



Cisco APIC NX-OS Style CLI Configuration Guide, Release 4.0(x)

First Published: 2018-09-24

Last Modified: 2019-07-30

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.

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)

© 2018 Cisco Systems, Inc. All rights reserved.

Please send general FSF & GNU inquiries to gnu@gnu.org. There are also [other ways to contact](#) the FSF. Please send broken links and other corrections or suggestions to bug-bash@gnu.org. Please see the [Translations README](#) for information on coordinating and submitting translations of this article.

Copyright © 2007, 2009, 2011 Free Software Foundation, Inc. Verbatim copying and distribution of this entire article are permitted worldwide, without royalty, in any medium, provided this notice, and the copyright notice, are preserved. Updated: Date: 2011/06/28 02:44:32

© 2018 Cisco Systems, Inc. All rights reserved.



CONTENTS

Full Cisco Trademarks with Software License ?

PREFACE

Preface **xxi**

Audience **xxi**

New and Changed Information **xxi**

Document Conventions **xxii**

Related Documentation **xxiii**

Documentation Feedback **xxiv**

CHAPTER 1

Using the APIC CLI **1**

Accessing the NX-OS Style CLI **1**

Using the NX-OS Style CLI for APIC **2**

Differences in Usage from NX-OS **5**

Mixing the NX-OS Style CLI and the APIC GUI **5**

About the Modes of Configuring Layer 3 External Connectivity **6**

CHAPTER 2

Configuring Fabric and Interfaces **9**

Fabric and Interface Configuration **9**

Graceful Insertion and Removal (GIR) Mode **10**

Removing a Switch to Maintenance Mode Using the CLI **11**

Inserting a Switch to Operation Mode Using CLI **11**

Configuring Physical Ports in Leaf Nodes and FEX Devices Using the NX-OS CLI **12**

Configuring Port Channels in Leaf Nodes and FEX Devices Using the NX-OS CLI **15**

Configuring Virtual Port Channels in Leaf Nodes and FEX Devices Using the NX-OS CLI **21**

Configuring FEX Connections Using Profiles with the NX-OS Style CLI **26**

Reflective Relay (802.1Qbg) **27**

Enabling Reflective Relay Using the NX-OS CLI	27
Configuring Policy Groups for Interfaces	29
Configuring Overrides for Interfaces	31
About Forwarding Error Correction	33
Configuring FEC Using NX-OS Style CLI	34

CHAPTER 3

Cisco ACI Smart Licensing	35
About Smart Licensing	35
Smart Licensing Usage Guidelines and Limitations	36
Pre-Registration Verifications	36
Verification Checklist for CSSM Configurations	36
Verification Checklist for Smart Licensing and APIC Configurations	36
Registering for Smart Licensing Using the CLI	36

CHAPTER 4

Configuring APIC High Availability	41
About Cold Standby for APIC Cluster	41
Switching Over Active APIC with Standby APIC Using CLI	42

CHAPTER 5

Configuring Tenants	43
Creating a Tenant, VRF, and Bridge Domain	43
Additional Bridge Domain Configuration	46
Configuring an Enforced Bridge Domain	48
Configuring an Enforced Bridge Domain	48
Configuring an Enforced Bridge Domain Using the NX-OS Style CLI	49
Creating an Application Endpoint Group	50
Configuring Legacy Forwarding Mode in the Bridge Domain	53
Configuring Contracts	55
Contract Inheritance	59
About Contract Inheritance	59
Configuring Application or uSeg EPG Contract Inheritance Using the NX-OS Style CLI	60
Configuring L2Out EPG Contract Inheritance Using the NX-OS Style CLI	63
Configuring External L3Out EPG Contract Inheritance Using the NX-OS Style CLI	66
Configuring Contract Preferred Groups	68
About Contract Preferred Groups	68

Configuring Contract Preferred Groups Using the NX-OS Style CLI	69
Exporting a Contract to Another Tenant	71
Configuring Contract or Subject Exceptions	73
Configuring Contract or Subject Exceptions for Contracts	73
Configure a Contract or Subject Exception Using the NX-OS Style CLI	74
Creating Quota Management	75
About APIC Quota Management Configuration	75
Creating a Quota Management Configuration Using the NX-OS Style CLI	75

CHAPTER 6

Configuring Layer 2 External Connectivity 77

Configuring Layer 2 External Connectivity	77
Configuring VLAN Domains	81
About VLAN Domains	81
Basic VLAN Domain Configuration	82
Advanced VLAN Domain Configuration	83
Associating a VLAN Domain to a Port	84
Associating a VLAN Domain to a Port-Channel	86
Associating a VLAN Domain to a Template Policy-Group	87
Associating a VLAN Domain to a Template Port-Channel	87
Associating a VLAN Domain to a Virtual Port-Channel	88
Configuring Q-in-Q Encapsulation Mapping for EPGs	89
Q-in-Q Encapsulation Mapping for EPGs	89
Mapping EPGs to Q-in-Q Encapsulated Leaf Interfaces Using the NX-OS Style CLI	90
Support Fibre Channel over Ethernet Traffic on the ACI Fabric	91
Supporting Fibre Channel over Ethernet Traffic on the ACI Fabric	91
FCoE NX-OS Style CLI Configuration	94
Configuring FCoE Connectivity Without Policies or Profiles Using the NX-OS Style CLI	94
Configuring FCoE Connectivity With Policies and Profiles Using the NX-OS Style CLI	97
Configuring FCoE Over FEX Using NX-OS Style CLI	100
Verifying FCoE Configuration Using the NX-OS Style CLI	102
Undeploying FCoE Elements Using the NX-OS Style CLI	102
Fibre Channel NPV	104
Fibre Channel Connectivity Overview	104
Fibre Channel N-Port Virtualization Guidelines and Limitations	106

Configuring FC Connectivity Without Policies or Profiles Using the NX-OS CLI	107
Configuring FC Connectivity With Policies or Profiles Using the NX-OS CLI	108
Configuring 802.1Q Tunnels	110
About ACI 802.1Q Tunnels	110
Configuring 802.1Q Tunnels Using the NX-OS Style CLI	112
Example: Configuring an 802.1Q Tunnel Using Ports with the NX-OS Style CLI	114
Example: Configuring an 802.1Q Tunnel Using Port-Channels with the NX-OS Style CLI	114
Example: Configuring an 802.1Q Tunnel Using Virtual Port-Channels with the NX-OS Style CLI	115
Configuring Dynamic Breakout Ports	116
Configuration of Dynamic Breakout Ports	116
Configuring Dynamic Breakout Ports Using the NX-OS Style CLI	116
Configuring Port Profiles	120
Configuring Port Profiles	120
Port Profile Configuration Summary	123
Configuring a Port Profile Using the NX-OS Style CLI	125
Verifying Port Profile Configuration and Conversion Using the NX-OS Style CLI	125
Microsegmentation on Virtual Switches	126
Configuring Microsegmentation on Virtual Switches	126
Configuring Microsegmentation with Cisco ACI Using the NX-OS-Style CLI	127
Configuring Microsegmentation on Bare-Metal	128
Using Microsegmentation with Network-based Attributes on Bare Metal	128
Configuring a Network-Based Microsegmented EPG in a Bare-Metal Environment Using the NX-OS Style CLI	129
Configuring Layer 2 IGMP Snoop Multicast	131
About Cisco APIC and IGMP Snooping	131
Enabling IGMP Snooping Static Port Groups	132
Configuring and Assigning an IGMP Snooping Policy to a Bridge Domain using the NX-OS Style CLI	132
Enabling IGMP Snooping and Multicast on Static Ports in the NX-OS Style CLI	133
Enabling IGMP Snoop Access Groups	134
Enabling Group Access to IGMP Snooping and Multicast using the NX-OS Style CLI	135
Deploying an EPG on a Specific Port with APIC Using the NX-OS Style CLI	137
Configuring Port Security	138
About Port Security and ACI	138

Port Security Guidelines and Restrictions	138
Port Security at Port Level	138
Configuring a Port Security Policy Group Template	138
Configuring Port Security on an Interface Using a Template	140
Configuring Port Security on an Interface Using Overrides	141
802.1x Port and Node Authentication	142
802.1x Port and Node Authentication	142
Configuring a Port Authentication Policy	142
Configuring a Node Authentication Policy	144
Configuring Proxy ARP	145
About Proxy ARP	145
Guidelines and Limitations	151
Configuring Proxy ARP Using the Cisco NX-OS Style CLI	151
Configuring Flood in Encapsulation	153
Configuring Traffic Storm Control	154
About Traffic Storm Control	154
Storm Control Guidelines	154
Configuring a Traffic Storm Control Policy Using the NX-OS Style CLI	155
Configuring MACsec	156
About MACsec	156
Guidelines and Limitations for MACsec	157
Configuring MACsec Using the NX-OS Style CLI	159
CHAPTER 7	
Configuring Layer 3 External Connectivity	163
About the Modes of Configuring Layer 3 External Connectivity	163
Configuring Layer 3 External Connectivity	165
Routed Connectivity to External Networks	165
About Routed Connectivity to Outside Networks	165
Layer 3 Out for Routed Connectivity to External Networks	166
Guidelines for Routed Connectivity to Outside Networks	168
External Layer 3 Outside Connection Types	170
Configuring a Layer 3 Outside for Tenant Networks Using the NX-OS Style CLI	173
NX-OS Style CLI Example: L3Out Prerequisites	176
NX-OS Style CLI Example: L3Out	176

Layer 3 Routed and Sub-Interface Port Channels	178
About Layer 3 Port Channels	178
Configuring a Layer 3 Routed Port-Channel Using the NX-OS CLI	178
Configuring a Layer 3 Sub-Interface Port-Channel Using the NX-OS CLI	181
Adding Ports to the Layer 3 Port-Channel Using the NX-OS CLI	184
Layer 3 Out to Layer 3 Out Inter-VRF Leaking	185
Configuring Shared Layer 3 Out Inter-VRF Leaking Using the NX-OS Style CLI - Named Example	186
Configuring Shared Layer 3 Out Inter-VRF Leaking Using the NX-OS Style CLI - Implicit Example	187
About SVI External Encapsulation Scope	189
Encapsulation Scope Syntax	191
Configuring SVI Interface Encapsulation Scope Using NX-OS Style CLI	191
About SVI Auto State	192
Guidelines and Limitations for SVI Auto State Behavior	193
Configuring SVI Auto State Using NX-OS Style CLI	193
Configuring an Interface and Static Route	194
OSPF Configuration	197
Configuring OSPF	197
Creating OSPF VRF and Interface Templates	201
BGP Configuration	205
Configuring BGP	205
Creating BGP Address Family and Timer Templates	206
Configuring BGP Address Family and Timers	207
Configuring a BGP Neighbor	209
Configuring a Per VRF Per Node BGP Timer Policy Using the NX-OS Style CLI	213
Configuring BGP Max Path	215
Configuring AS Path Prepend	216
Configuring AS Path Prepend Using the NX-OS Style CLI	216
Route Distribution Into BGP	217
Configuring a Route-Profile with Tenant Scope	217
Configuring a Redistribute Route-Profile	218
Configuring BGP Route Dampening	220
EIGRP Configuration	223

Creating EIGRP VRF and Interface Templates	223
Configuring EIGRP Address Family and Counters	225
Configuring an EIGRP Interface	227
Configuring Route-Maps	230
Configuring Templates	230
About Route Profiles	230
Configuring a Tenant-Scoped Route Profile	230
Configuring a VRF-Scoped Route Profile	232
Creating a Route-Map	235
Configuring Route-Maps in Routing Protocols	239
Configuring an Export Map (Inter-VRF Route Leak)	240
Configuring Bi-Directional Route Forwarding (BFD)	242
About BFD	242
Configuring BFD Globally	242
Configuring BFD Globally on Leaf Switch Using the NX-OS Style CLI	245
Configuring BFD Globally on Spine Switch Using the NX-OS Style CLI	246
Overriding Global BFD Settings	247
Configuring BFD Interface Override Policy	247
Applying the BFD Interface Override Policy to Interfaces	250
Enabling BFD on Consumer Protocols	252
Enabling BFD on the BGP Consumer Protocol	252
Enabling BFD on the EIGRP Consumer Protocol	253
Enabling BFD on the OSPF Consumer Protocol	254
Enabling BFD on the Static Route Consumer Protocol	255
Configuring BFD Consumer Protocols Using the NX-OS Style CLI	256
Configuring Layer 3 Multicast	257
Layer 3 Multicast	257
Guidelines and Restrictions for Configuring Layer 3 Multicast	258
Configuration Steps for Layer 3 Multicast	260
Configuring PIM Options for Layer 3 Multicast	260
Configuring IGMP Options on the VRF for Layer 3 Multicast	263
Configuring an L3 Out for Layer 3 Multicast	268
Example: Configuring Layer 3 Multicast	272
Configuring External-L3 EPGs	272

Configuring Layer 3 External Connectivity Using the Named Mode	275
Creating a Named L3Out	275
Configuring Layer 3 Interfaces for a Named L3Out	277
Configuring Route Maps for a Named L3Out	279
Configuring Routing Protocols for a Named L3Out	283
Configuring BGP for a Named L3Out	283
Configuring OSPF for a Named L3Out	285
Configuring EIGRP for a Named L3Out	287
Configuring External-L3 EPGs for a Named L3Out	290
IPv6 Neighbor Discovery	291
Neighbor Discovery	291
Configuring a Tenant, VRF, and Bridge Domain with IPv6 Neighbor Discovery on the Bridge Domain Using the NX-OS Style CLI	292
Guidelines and Limitations	293
Configuring an IPv6 Neighbor Discovery Interface Policy with RA on a Layer 3 Interface Using the NX-OS Style CLI	293
Microsoft NLB	296
Configuring Microsoft NLB in Unicast Mode Using the NX-OS Style CLI	296
Configuring Microsoft NLB in Multicast Mode Using the NX-OS Style CLI	297
Configuring Microsoft NLB in IGMP Mode Using the NX-OS Style CLI	298
MLD Snooping	299
Configuring and Assigning an MLD Snooping Policy to a Bridge Domain using the NX-OS Style CLI	299
Configuring HSRP	302
Configuring HSRP in Cisco APIC Using Inline Parameters in NX-OS Style CLI	302
Configuring HSRP in Cisco APIC Using Template and Policy in NX-OS Style CLI	303
Cisco ACI GOLF	305
Cisco ACI GOLF	305
Configuration Tasks to Configure Cisco ACI GOLF Services Using the NX-OS Style CLI	307
Configuring a Spine and the Infra Tenant for BGP EVPN, Using the NX-OS Style CLI	308
APIC GOLF Connections Shared by Multi-Site Sites	310
Recommended Shared GOLF Configuration Using the NX-OS Style CLI	311
Configuring BGP to Support BGP EVPN on a Spine, Using the NX-OS Style CLI	312
Configuring a Tenant for BGP EVPN Using the NX-OS Style CLI	313
Configuring a Route Map	314

Enabling Distributing BGP EVPN Type-2 Host Routes to a DCIG Using the NX-OS Style CLI	317
Cisco ACI GOLF Configuration Example, Using the NX-OS Style CLI	317
Troubleshooting EVPN Type-2 Route Distribution to a DCIG	319
Multipod_Fabric	322
About Multipod Fabric	322
Assigning Switches in a Multipod Fabric	322
Configuring Fabric-External Connectivity for a Multipod Fabric	323
Configuring Spine Interfaces and OSPF for a Multipod Fabric	325
Remote Leaf Switches	328
About Remote Leaf Switches in the ACI Fabric	328
Remote Leaf Switch Hardware Requirements	330
Restrictions and Limitations	331
WAN Router and Remote Leaf Switch Configuration Guidelines	332
Configure Remote Leaf Switches Using the NX-OS Style CLI	333
Transit Routing	336
Transit Routing in the ACI Fabric	336
Transit Routing Related Topics	337
Transit Routing Overview	337
Guidelines for Transit Routing	338
Configure Transit Routing Using the NX-OS Style CLI	343
Example: Transit Routing	346
CHAPTER 8	
Configuring Cisco ACI QoS	351
L3Outs QoS	351
Configuring QoS Contract for L3Out Using NX-OS Style CLI	351
Configuring QoS Directly on L3Out Using CLI	352
CoS Preservation	353
Class of Service (CoS) Preservation for Ingress and Egress Traffic	353
Enable Class Of Service (CoS) Preservation Using NX-OS Style CLI	353
Multipod QoS	354
Creating DSCP Translation Policy Using NX-OS Style CLI	354
Multipod QoS and DSCP Translation Policy	355
Translating QoS Ingress Markings to Egress Markings	356
Translating Ingress to Egress QoS Markings	356

Creating Custom QoS Policy Using NX-OS Style CLI 356

CHAPTER 9

Configuring Management Interfaces 359

Configuring Out-of-Band Management Access 359

Configuring Inband Management Access 362

Configuring Inband Management Access to a Switch from an Outside Network 362

Configuring Inband Management Access to a Controller from an Outside Network 363

Configuring Inband Management Connectivity to the Management Station 366

Configuring Inband Management Contract to Open HTTPS/SSH Ports 368

CHAPTER 10

Configuring Security 371

About Security Configuration 371

Configuring AAA 372

Configuring Security Servers 376

Configuring a RADIUS Server 376

Configuring a TACACS+ Server 378

Configuring an LDAP Server 380

Configuring the Password Policy 383

Configuring Users 385

Configuring a Locally Authenticated User 385

Configuring a Certificate and SSH-Key for a Local User 388

Configuring Public Key Infrastructure 390

Configuring a Certificate Authority and Chain of Trust 390

Configuring Keys and a Keyring 390

Generating a Certificate Signing Request 392

Configuring Webtokens 394

Configuring Communication Policies 396

Configuring the HTTP Policy 396

Configuring the HTTPS Policy 397

Configuring the SSH Policy 399

Configuring the Telnet Policy 400

Configuring AES Encryption 401

Configuring Fabric Secure Mode 402

Configuring COOP Authentication 404

About COOP Authentication	404
Configuring COOP Authentication	404
Configuring FIPS	405
About Federal Information Processing Standards (FIPS)	405
Guidelines and Limitations	405
Configuring FIPS for Cisco APIC Using NX-OS Style CLI	406
Configuring Control Plane Policing	406
Information About CoPP	406
Guidelines and Limitations for CoPP	408
Configuring CoPP Using the Cisco NX-OS CLI	409
Configuring Per Interface Per Protocol CoPP Policy Using the NX-OS Style CLI	409
Configuring First Hop Security	410
About First Hop Security	410
ACI FHS Deployment	411
Guidelines and Limitations	411
Configuring FHS Using the NX-OS CLI	411
Configuring 802.1x	417
802.1X Overview	417
Host Support	417
Authentication Modes	418
Guidelines and Limitations	418
Configuration Overview	419
Configuring 802.1X Node Authentication Using NX-OS Style CLI	419
Configuring 802.1X Port Authentication Using the NX-OS Style CLI	420

CHAPTER 11
Configuring Anycast Services 423

About Anycast Services 423

Configuring Anycast Services Using the NX-OS Style CLI 423

CHAPTER 12
Configuring VMM 427

Configuring VMM 427

CHAPTER 13
Configuring Layer 4 to Layer 7 Services 429

Configuring Layer 4 to Layer 7 Services 429

CHAPTER 14**Configuring Global Policies 431**

- About Global Policies 431
- Configuring Out-of-Band Management NTP 431
- Configuring the System Clock 434
- Configuring Error Disable Recovery 435
- Configuring Link Level Discovery Protocol 436
- Configuring Miscabling Protocol 437
- Configuring the Endpoint Loop Protection Policy 439
- Configuring the Rogue Endpoint Control Policy 440
 - About the Rogue Endpoint Control Policy 440
 - Configure Rogue Endpoint Control Using the NX-OS Style CLI 440
- Configuring IP Aging 441
 - Overview 441
 - Configuring the IP Aging Policy Using the NX-OS-Style CLI 441
- Configuring the Dynamic Load Balancer 442
- Configuring Spanning Tree Protocol 443
- Configuring IS-IS 445
- Configuring BGP Route Reflectors 447
- Decommissioning a Node 448
- Configuring Power Management 449
- Configuring a Scheduler 451
- Configuring System MTU 453
- About PTP 454
 - Guidelines and Limitations 456
 - Configuring PTP Using the NX-OS CLI 457

CHAPTER 15**Configuring Cisco Tetration Analytics 461**

- Overview 461
- Configuring Cisco Tetration Analytics Using the NX-OS Style CLI 461

CHAPTER 16**Configuring NetFlow 465**

- About NetFlow 465

Configuring a NetFlow Exporter Policy for Virtual Machine Networking Using the NX-OS-Style CLI	466
Configuring NetFlow and Tetration Analytics Feature Priority Through Node Control Policy Using NX-OS-Style CLI	466
Configuring NetFlow Node Policy Using the NX-OS-Style CLI	467
Configuring NetFlow Infra Selectors Using the NX-OS-Style CLI	467
Configuring NetFlow Overrides Using the NX-OS-Style CLI	469
Configuring NetFlow Tenant Hierarchy Using the NX-OS-Style CLI	470
Consuming a NetFlow Exporter Policy Under a VMM Domain Using the NX-OS-Style CLI for VMware VDS	473
Enabling or Disabling NetFlow on an Endpoint Group Using the NX-OS-Style CLI for VMware VDS	474

CHAPTER 17

Managing Firmware 475

Managing Firmware	475
Adding or Removing Repository Images	475
Changing Catalog Firmware	476
Upgrading Controller Firmware	477
Upgrading Switch Firmware	479

CHAPTER 18

Managing the Configuration with Snapshots 483

About Configuration Management and Snapshots	483
Exporting a Snapshot	483
Importing a Snapshot	485
Rollback Configuration Using Snapshots	486
Uploading or Downloading a Snapshot File to a Remote Path	488
Managing Snapshot Files and Jobs	489

CHAPTER 19

Configuring Monitoring 491

Configuring Syslog	491
Configuring a Logging Server Group	491
Configuring Syslog	493
Configuring Call Home	494
Configuring the Call Home Policy	494

Configuring a Call Home Destination Profile	496
Call Home Destination Profile Configuration Commands	499
Configuring a Call Home Query	499
Query Subtree Categories	501
Configuring TACACS External Logging	502
Creating a TACACS External Logging Destination Group Using the NX-OS-Style CLI	502
Creating a TACACS External Logging Source Using the NX-OS-Style CLI	503
Sending an On-Demand Techsupport File Using the NX-OS Style CLI	504
Configuring a Remote Path for File Export	505
Using Show Commands for Monitoring	507
About Using the Show Commands	507
Using the show faults Command	507
Using the show events Command	508
Using the show health Command	509
Using the show audits Command	510
Using the show stats Command	511
Entity Filters for Show Commands	512
Configuring SNMP	513
Configuring SNMP Policy Using CLI	514
Configuring Smart Callhome	516
About Smart Callhome	516
Creating a Smart Callhome Destination Group Using the NX-OS-Style CLI	516

CHAPTER 20
Configuring SPAN 519

Configuring SPAN and ERSPAN	519
SPAN Guidelines and Restrictions	519
Configuring Local SPAN in Access Mode	521
Configuring ERSPAN in Access Mode	523
Configuring ERSPAN in Fabric Mode	526
Configuring ERSPAN in Tenant Mode	529

CHAPTER 21
Applying the show running config Output to Another Cisco APIC 533

About Import and Export Configurations	533
Import and Export Configuration Guidelines and Limitations	533

Exporting a CLI Configuration 533

Importing a CLI Configuration 534

CHAPTER 22

Configuring a Forwarding Scale Profile Policy 537

Forwarding Scale Profile Policy Overview 537

Supported Platforms for Forwarding Scale Profile Policies 539

Configuring the Forwarding Scale Profile Policy Using the NX-OS-Style CLI 539

APPENDIX A

Verified Scalability Using the CLI 543

CLI Scalability Limits 543

APPENDIX B

Use Case: Three-Tier Application with Transit Topology 545

About Deploying a Three-Tier Application with Transit Topology 545

Deploying a Three-Tier Application 547

Transit Routing with OSPF and BGP 549

APPENDIX C

Examples: Show Commands 551

Examples: Show Commands 551



Preface

- [Audience, on page xxi](#)
- [New and Changed Information, on page xxi](#)
- [Document Conventions, on page xxii](#)
- [Related Documentation, on page xxiii](#)
- [Documentation Feedback, on page xxiv](#)

Audience

This guide is intended for network and systems administrators who configure and maintain the Application Centric Infrastructure fabric.

New and Changed Information

The following table provides an overview of the significant changes to this guide up to the 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 APIC, Release 4.0(x)

Feature	Description	Where Documented
Service graph support	EPGs created by service graphs can be included in contract preferred groups.	Configuring Tenants, on page 43
Breakout limitation on N9K-C93240YC-FX2 switches	Breakout is not supported on converted downlinks.	Configuring Layer 2 External Connectivity, on page 77

Feature	Description	Where Documented
Change in behavior for remote leaf switches in the ACI fabric	There are several changes in behavior for remote leaf switches in the ACI fabric with this release, including a reduction of WAN bandwidth use by decoupling services from the spine-proxy and the resolution of unknown L3 endpoints (through ToR glean process) in a remote leaf site when the spine-proxy is not reachable.	Configuring Layer 3 External Connectivity, on page 163
Support for L3Out QoS configuration directly on the L3Out interfaces.	Starting with Release 4.0(1), L3Out QoS can be configured directly on the L3Out interfaces.	Configuring Cisco ACI QoS, on page 351

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.
<code>variable</code>	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.

Convention	Description
boldface screen font	Information you must enter is in boldface screen font.
<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

Using the APIC CLI

- [Accessing the NX-OS Style CLI, on page 1](#)
- [Using the NX-OS Style CLI for APIC, on page 2](#)
- [Differences in Usage from NX-OS, on page 5](#)
- [Mixing the NX-OS Style CLI and the APIC GUI, on page 5](#)

Accessing the NX-OS Style CLI



Note From Cisco APIC Release 1.0 until Release 1.2, the default CLI was a Bash shell with commands to directly operate on managed objects (MOs) and properties of the Management Information Model. Beginning with Cisco APIC Release 1.2, the default CLI is a NX-OS style CLI. The object model CLI is available by typing the **bash** command at the initial CLI prompt.

- Step 1** From a secure shell (SSH) client, open an SSH connection to APIC at *username@ip-address*.
Use the administrator login name and the out-of-band management IP address that you configured during the initial setup. For example, `admin@192.168.10.1`.
- Step 2** When prompted, enter the administrator password.

What to do next

When you enter the NX-OS style CLI, the initial command level is the EXEC level. From this level, you can reach these configuration modes:

- To continue in the NX-OS style CLI, you can stay in EXEC mode or you can type **configure** to enter global configuration mode.
For information about NX-OS style CLI commands, see the *Cisco APIC NX-OS Style CLI Command Reference*.
- To reach the object model CLI, type **bash**.

For information about object mode CLI commands, see the *Cisco APIC Command-Line Interface User Guide, APIC Releases 1.0 and 1.1*.

Using the NX-OS Style CLI for APIC

Using CLI Command Modes

The NX-OS style CLI is organized in a hierarchy of command modes with EXEC mode as the root, containing a tree of configuration submodes beginning with global configuration mode. The commands available to you depend on the mode you are in. To obtain a list of available commands in any mode, type a question mark (?) at the system prompt.

This table lists and describes the two most commonly used modes (EXEC and global configuration) along with an example submode (DNS). The table shows how to enter and exit the modes, and the resulting system prompts. The system prompt helps to identify which mode you are in and the commands that are available to you in that mode.

Mode	Access Method	Prompt	Exit Method
EXEC	From the APIC prompt, enter execsh .	apic#	To exit to the login prompt, use the exit command.
Global configuration	From EXEC mode, enter the configure command.	apic(config)#	To exit from a configuration submode to its parent mode, use the exit command.
DNS configuration	From global configuration mode, enter the dns command.	apic(config-dns)#	To exit from any configuration mode or submode to EXEC mode, use the end command.

CLI Command Hierarchy

Configuration mode has several submodes, with commands that perform similar functions grouped under the same level. For example, all commands that display information about the system, configuration, or hardware are grouped under the **show** command, and all commands that allow you to configure the switch are grouped under the **configure** command.

To execute a command that is not available in EXEC mode, you navigate to its submode starting at the top level of the hierarchy. For example, to configure DNS settings, use the **configure** command to enter the global configuration mode, then enter the **dns** command. When you are in the DNS configuration submode, you can query the available commands. as in this example:

```
apic1# configure
apic1(config)# dns
apic1(config-dns)# ?
address    Configure the ip address for dns servers
domain     Configure the domains for dns servers
exit       Exit from current mode
fabric     Show fabric related information
no         Negate a command or set its defaults
show       Show running system information
use-vrf    Configure the management vrf for dns servers
where      Show the current mode
```

```
apic1(config-dns) # end
apic1#
```

Each submode places you further down in the prompt hierarchy. To view the hierarchy for the current mode, use the **configure** command, as shown in this example:

```
apic1# configure
apic1(config) # bgp-fabric
apic1(config-bgp-fabric) # where
configure t; bgp-fabric
apic1(config-bgp-fabric) #
```

To leave the current level and return to the previous level, type **exit**. To return directly to the EXEC level, type **end**.

EXEC Mode Commands

When you start a CLI session, you begin in EXEC mode. From EXEC mode, you can enter configuration mode. Most EXEC commands are one-time commands, such as show commands, which display the current configuration status.

Configuration Mode Commands

Configuration mode allows you to make changes to the existing configuration. When you save the configuration, these commands are saved across switch reboots. Once you are in configuration mode, you can enter a variety of protocol-specific modes. Configuration mode is the starting point for all configuration commands.

Listing Commands and Syntax

In any command mode, you can obtain a list of available commands by entering a question mark (?).

```
apic1(config-dns) # ?
address    Configure the ip address for dns servers
domain     Configure the domains for dns servers
exit       Exit from current mode
fabric     Show fabric related information
no         Negate a command or set its defaults
show       Show running system information
use-vrf    Configure the management vrf for dns servers
where      Show the current mode

apic1(config-dns) # end
apic1#
```

To see a list of commands that begin with a particular character sequence, type those characters followed by a question mark (?). Do not include a space before the question mark.

```
apic1(config) # sh ?
aaa          Show AAA information
access-list  Show Access-list Information
accounting   Show accounting information
acllog       Show acllog information
. . .
```

To complete a command after you begin typing, type a tab.

```
apicl# qu<TAB>
apicl# quota
```

To list keywords or arguments, enter a question mark in place of a keyword or argument. Include a space before the question mark. This form of help is called command syntax help because it reminds you which keywords or arguments are applicable based on the commands, keywords, and arguments you have already entered.

```
apicl(config-dns)# use-vrf ?
inband-mgmt  Configure dns on inband
oob-mgmt     Configure dns on out-of-band

apicl(config-dns)#
```

You can also abbreviate a command if the abbreviation is unambiguous. In this example, the **configure** command is abbreviated.

```
apicl# conf
apicl(config)#
```

Undoing or Reverting to Default Values or Conditions Using the 'no' Prefix

For many configuration commands, you can precede the command with the **no** keyword to remove a setting or to restore a setting to the default value. This example shows how to remove a previously-configured DNS address from the configuration.

```
apicl(config-dns)# address 192.0.20.123 preferred
apicl(config-dns)# show dns-address
Address                Preferred
-----
192.0.20.123          yes

apicl(config-dns)# no address 192.0.20.123
apicl(config-dns)# show dns-address
Address                Preferred
-----
```

Executing BASH Commands From the NX-OS Style CLI

To execute a single command in the bash shell, type **bash -c 'path/command'** as shown in this example.

```
apicl# bash -c '/controller/sbin/acidiag avread'
```

You can execute a bash command from any mode or submode in the NX-OS style CLI.

Entering Configuration Text with Spaces or Special Characters

When a configuration field consists of user-defined text, special characters such as '\$' should be escaped ('\$') or the entire word or string should be wrapped in single quotes to avoid misinterpretation by Bash.

Differences in Usage from NX-OS

The usage of the NX-OS style CLI for APIC differs from the traditional NX-OS CLI in these ways:

- Global configuration mode is entered with the **configure** command instead of **configure terminal**.
- To perform node-level configuration on a particular leaf switch, you must first navigate to that switch using the **leaf** command.
- The command syntax for specifying a physical port is slightly different. For example, an Ethernet port is specified as **eth x/y** instead of **ethx/y**.
- When a configuration field consists of user-defined text, such as a password, special characters such as '\$' or '!' should be escaped with a backslash ('\') or the entire word or string should be wrapped in single quotes to avoid misinterpretation by Bash.
- Some command shortcuts are different due to Bash behavior:
 - **Ctrl-D** exits a session.
 - **Ctrl-Z** suspends a job.
- OSPF configuration adds **area route-map** and **area connectivity** commands.

Mixing the NX-OS Style CLI and the APIC GUI

Basic mode is deprecated since Cisco APIC Release 3.0(1). There is only one GUI as of that release.

**Caution**

Configurations done through the NX-OS style CLI are rendered in the APIC GUI. They can be seen, but sometimes may not be editable in the GUI. Also changes made in the APIC GUI may be seen in the NX-OS style CLI, but may only partially work. See the following examples:

- Do not mix the GUI and the CLI, when doing per-interface configuration on APIC. Configurations performed in the GUI, may only partially work in the NX-OS CLI.

For example, if you configure a switch port in the GUI at **Tenants > *tenant-name* > Application Profiles > *application-profile-name* > Application EPGs > *EPG-name* > Static Ports > Deploy Static EPG on PC, VPC, or Interface**

Then you use the show running-config command in the NX-OS style CLI, you receive output such as:

```
leaf 102
interface ethernet 1/15
switchport trunk allowed vlan 201 tenant t1 application ap1 epg ep1
exit
exit
```

If you use these commands to configure a static port in the NX-OS style CLI, the following error occurs:

```
apic1(config)# leaf 102
apic1(config-leaf)# interface ethernet 1/15
apic1(config-leaf-if)# switchport trunk allowed vlan 201 tenant t1 application ap1 epg
ep1
No vlan-domain associated to node 102 interface ethernet1/15 encap vlan-201
```

This occurs because the CLI has validations that are not performed by the APIC GUI. For the commands from the show running-config command to function in the NX-OS CLI, a vlan-domain must have been previously configured. The order of configuration is not enforced in the GUI.

For the steps to remove such objects, see *Troubleshooting Unwanted _ui_ Objects* in the *APIC Troubleshooting Guide*.

About the Modes of Configuring Layer 3 External Connectivity

Because APIC supports multiple user interfaces (UIs) for configuration, the potential exists for unintended interactions when you create a configuration with one UI and later modify the configuration with another UI. This section describes considerations for configuring Layer 3 external connectivity with the APIC NX-OS style CLI, when you may also be using other APIC user interfaces.

When you configure Layer 3 external connectivity with the APIC NX-OS style CLI, you have the choice of two modes:

- Implicit mode, a simpler mode, is not compatible with the APIC GUI or the REST API.
- Named (or Explicit) mode is compatible with the APIC GUI and the REST API.

In either case, the configuration should be considered read-only in the incompatible UI.

How the Modes Differ

In both modes, the configuration settings are defined within an internal container object, the "L3 Outside" (or "L3Out"), which is an instance of the **l3extOut** class in the API. The main difference between the two modes is in the naming of this container object instance:

- Implicit mode—the naming of the container is implicit and does not appear in the CLI commands. The CLI creates and maintains these objects internally.
- Named mode—the naming is provided by the user. CLI commands in the Named Mode have an additional **l3Out** field. To configure the named L3Out correctly and avoid faults, the user is expected to understand the API object model for external Layer 3 configuration.

**Note**

Except for the procedures in the *Configuring Layer 3 External Connectivity Using the Named Mode* section, this guide describes Implicit mode procedures.

Guidelines and Restrictions

- In the same APIC instance, both modes can be used together for configuring Layer 3 external connectivity with the following restriction: The Layer 3 external connectivity configuration for a given combination of tenant, VRF, and leaf can be done only through one mode.
- For a given tenant VRF, the policy domain where the External-l3 EPG can be placed can be in either the Named mode or in the Implicit mode. The recommended configuration method is to use only one mode for a given tenant VRF combination across all the nodes where the given tenant VRF is deployed for Layer 3 external connectivity. The modes can be different across different tenants or different VRFs and no restrictions apply.
- In some cases, an incoming configuration to a Cisco APIC cluster will be validated against inconsistencies, where the validations involve externally-visible configurations (northbound traffic through the L3Outs). An Invalid Configuration error message will appear for those situations where the configuration is invalid.
- The external Layer 3 features are supported in both configuration modes, with the following exception:
 - Route-peering and Route Health Injection (RHI) with a L4-L7 Service Appliance is supported only in the Named mode. The Named mode should be used across all border leaf switches for the tenant VRF where route-peering is involved.
- Layer 3 external network objects (l3extOut) created using the Implicit mode CLI procedures are identified by names starting with “_ui_” and are marked as read-only in the GUI. The CLI partitions these external-l3 networks by function, such as interfaces, protocols, route-map, and EPG. Configuration modifications performed through the REST API can break this structure, preventing further modification through the CLI.

For the steps to remove such objects, see *Troubleshooting Unwanted _ui_ Objects* in the *APIC Troubleshooting Guide*.



CHAPTER 2

Configuring Fabric and Interfaces

- [Fabric and Interface Configuration](#), on page 9
- [Graceful Insertion and Removal \(GIR\) Mode](#), on page 10
- [Configuring Physical Ports in Leaf Nodes and FEX Devices Using the NX-OS CLI](#), on page 12
- [Configuring Port Channels in Leaf Nodes and FEX Devices Using the NX-OS CLI](#), on page 15
- [Configuring Virtual Port Channels in Leaf Nodes and FEX Devices Using the NX-OS CLI](#), on page 21
- [Configuring FEX Connections Using Profiles with the NX-OS Style CLI](#), on page 26
- [Reflective Relay \(802.1Qbg\)](#), on page 27
- [Configuring Policy Groups for Interfaces](#), on page 29
- [Configuring Overrides for Interfaces](#), on page 31
- [About Forwarding Error Correction](#), on page 33

Fabric and Interface Configuration

To form the ACI fabric, Cisco Nexus 9000 Series ACI-mode switches are deployed in a leaf and spine topology managed by the APIC controller. Each leaf node is connected to all spine nodes with no connectivity between the leaf nodes. The interconnecting links between the leaf and spine nodes are called fabric links and the respective ports are called fabric ports. The fabric ports do not require user configuration for normal operation as these are auto discovered and factory default configuration is applied during fabric bring-up. All endpoint devices are connected to the leaf nodes through access ports. The access ports must be configured similar to those in NX-OS switches. Both fabric and access ports are represented as Interfaces as in NX-OS.

The leaf and spine nodes are considered different objects in the ACI model and support different sets of policies. In the CLI, these nodes are represented as leaf and spine respectively while both are commonly referred to as nodes. Leaf and spine node values are unique across all the pods in the fabric. FEX modules, if attached to the leaf nodes, will have fex-id values unique only within each leaf. For example, two leaf nodes can each have a FEX 101 attached.



Note

Configuring FEX connections with FEX IDs 165 to 199 is not supported in the APIC GUI. To use one of these FEX IDs, configure the profile using the NX-OS style CLI. For more information, see *Configuring FEX Connections Using Interface Profiles with the NX-OS Style CLI*.

As of Cisco APIC, Release 3.0(1k), connections to FEX modules can be configured as profiles.

Interface Naming for Leaf and FEX Interfaces

In ACI fabric, most interface configuration is done for physical ports, port-channels, or vPCs (either directly connected to leaf nodes or connected through FEX modules). The general command syntax for each interface type is shown in the following table.

Interface Type	Command Syntax	Examples
Port	interface ethernet <i>slot/port</i>	interface eth 1/1
FEX Port	interface ethernet <i>fex-id/slot/port</i>	interface eth 101/1/1
Port-channel	interface port-channel <i>name</i>	interface port-channel foo
FEX Port-channel	interface port-channel <i>name</i> fex <i>fex-id</i>	interface port-channel foo flex 101
Virtual Port-channel (VPC)	interface vpc <i>name</i>	interface vpc foo
vPC over FEX	interface vpc <i>name</i> fex <i>fex-a</i> <i>fex-b</i>	interface vpc foo flex 101 102

Graceful Insertion and Removal (GIR) Mode

The Graceful Insertion and Removal (GIR) mode, or maintenance mode, allows you to isolate a switch from the network with minimum service disruption. In the GIR mode you can perform real-time debugging without affecting traffic.

You can use graceful insertion and removal 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. When you are finished performing the debugging operations, you can use graceful insertion to return the switch to its fully operational (normal) mode. 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 maintenance mode TOR does not attract traffic from the spine switches. In addition, all the front-panel interfaces are shutdown on the switch except the fabric interfaces. 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)

Important Notes

- Upgrading or downgrading a switch in maintenance mode is not supported.

- 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.
- For multi-pod, **IS-IS metric for redistributed routes** should be set to less than 63. 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.
- For a GIR upgrade, Cisco Application Policy Infrastructure Controller (Cisco APIC)-connected leaf switches must be put into different maintenance groups such that the Cisco APIC-connected leaf switches get upgraded one at a time.

Removing a Switch to Maintenance Mode Using the CLI

Use this procedure to remove a switch to maintenance mode using the CLI.

SUMMARY STEPS

1. `[no]debug-switch node_id or node_name`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>[no]debug-switch node_id or node_name</code>	Removes the switch to maintenance mode.

Inserting a Switch to Operation Mode Using CLI

Use this procedure to insert a switch to operational mode using the CLI.

SUMMARY STEPS

1. `[no]no debug-switch node_id or node_name`

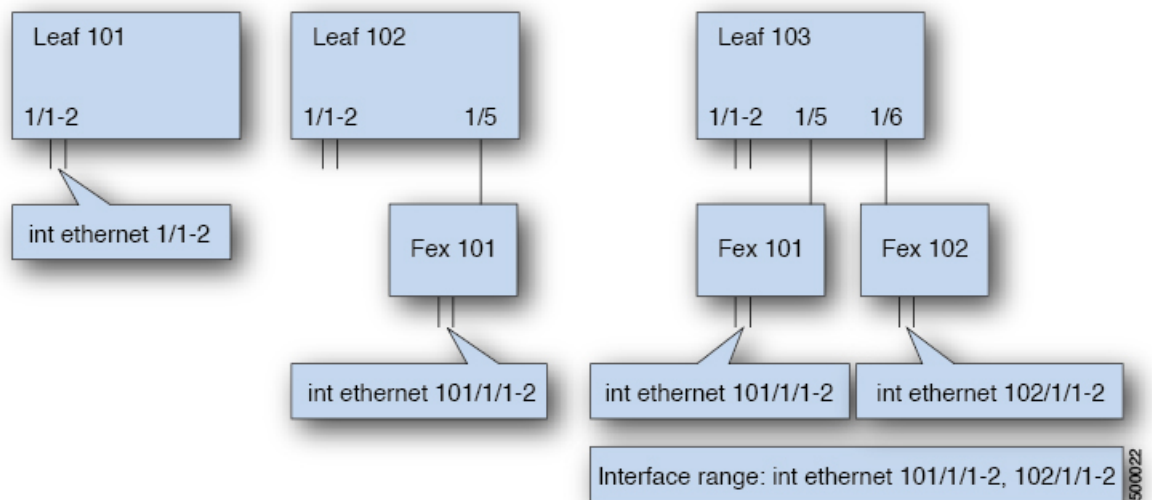
DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>[no]no debug-switch node_id or node_name</code>	Inserts the switch to operational mode.

Configuring Physical Ports in Leaf Nodes and FEX Devices Using the NX-OS CLI

The commands in the following examples create many managed objects (MOs) in the ACI policy model that are fully compatible with the REST API/SDK and GUI. However, the CLI user can focus on the intended network configuration instead of ACI model internals.

The following figure shows examples of Ethernet ports directly on leaf nodes or FEX modules attached to leaf nodes and how each is represented in the CLI. For FEX ports, the *fex-id* is included in the naming of the port itself as in **ethernet 101/1/1**. While describing an interface range, the **ethernet** keyword need not be repeated as in NX-OS. Example: **interface ethernet 101/1/1-2, 102/1/1-2**.



- Leaf node ID numbers are global.
- The *fex-id* numbers are local to each leaf.
- Note the space after the keyword **ethernet**.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **interface** *type*
4. (Optional) **fex** *associate node-id*
5. **speed** *speed*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 102</code>	Specifies the leaf or leaves to be configured. The <i>node-id</i> can be a single node ID or a range of IDs, in the form <i>node-id1–node-id2</i> , to which the configuration will be applied.
Step 3	interface <i>type</i> Example: <code>apic1(config-leaf)# interface ethernet 1/2</code>	Specifies the interface that you are configuring. You can specify the interface type and identity. For an Ethernet port, use “ethernet slot / port.”
Step 4	(Optional) fex associate <i>node-id</i> Example: <code>apic1(config-leaf-if)# flex associate 101</code>	If the interface or interfaces to be configured are FEX interfaces, you must use this command to attach the FEX module to a leaf node before configuration. Note This step is required before creating a port-channel using FEX ports.
Step 5	speed <i>speed</i> Example: <code>apic1(config-leaf-if)# speed 10G</code>	The speed setting is shown as an example. At this point you can configure any of the interface settings shown in the table below.

The following table shows the interface settings that can be configured at this point.

Command	Purpose
[no] shut	Shut down physical interface
[no] speed <i>speedValue</i>	Set the speed for physical interface
[no] link debounce time <i>time</i>	Set link debounce
[no] negotiate auto	Configure negotiate
[no] cdp enable	Disable/enable Cisco Discovery Protocol (CDP)
[no] mcp enable	Disable/enable Mis-cabling Protocol (MCP)
[no] lldp transmit	Set the transmit for physical interface
[no] lldp receive	Set the LLDP receive for physical interface
spanning-tree {bpduguard bpdufilter} {enable disable}	Configure spanning tree BPDU
[no] storm-control level <i>percentage</i> [burst-rate <i>percentage</i>]	Storm-control configuration (percentage)

Command	Purpose
[no] storm-control pps <i>packets-per-second</i> burst-rate <i>packets-per-second</i>	Storm-control configuration (packets-per-second)

Examples

Configure one port in a leaf node. The following example shows how to configure the interface eth1/2 in leaf 101 for the following properties: speed, cdp, and admin state.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/2
apic1(config-leaf-if)# speed 10G
apic1(config-leaf-if)# cdp enable
apic1(config-leaf-if)# no shut
```

Configure multiple ports in multiple leaf nodes. The following example shows the configuration of speed for interfaces eth1/1-10 for each of the leaf nodes 101-103.

```
apic1(config)# leaf 101-103
apic1(config-leaf)# interface eth 1/1-10
apic1(config-leaf-if)# speed 10G
```

Attach a FEX to a leaf node. The following example shows how to attach a FEX module to a leaf node. Unlike in NX-OS, the leaf port Eth1/5 is implicitly configured as fabric port and a FEX fabric port-channel is created internally with the FEX uplink port(s). In ACI, the FEX fabric port-channels use default configuration and no user configuration is allowed.



Note

This step is required before creating a port-channel using FEX ports, as described in the next example.

```
apic1(config)# leaf 102
apic1(config-leaf)# interface eth 1/5
apic1(config-leaf-if)# fex associate 101
```

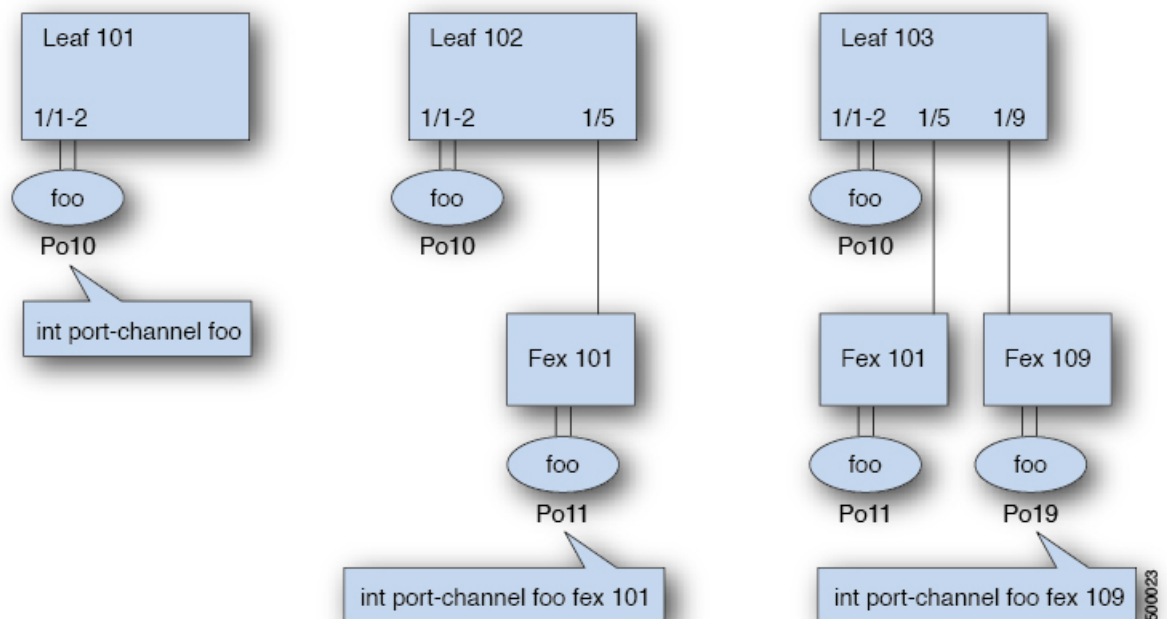
Configure FEX ports attached to leaf nodes. This example shows configuration of speed for interfaces eth1/1-10 in FEX module 101 attached to each of the leaf nodes 102-103. The FEX ID 101 is included in the port identifier. FEX IDs start with 101 and are local to a leaf.

```
apic1(config)# leaf 102-103
apic1(config-leaf)# interface eth 101/1/1-10
apic1(config-leaf-if)# speed 1G
```

Configuring Port Channels in Leaf Nodes and FEX Devices Using the NX-OS CLI

Port-channels are logical interfaces in NX-OS used to aggregate bandwidth for multiple physical ports and also for providing redundancy in case of link failures. In NX-OS, port-channel interfaces are identified by user-specified numbers in the range 1 to 4096 unique within a node. Port-channel interfaces are either configured explicitly (using the **interface port-channel** command) or created implicitly (using the **channel-group** command). The configuration of the port-channel interface is applied to all the member ports of the port-channel. There are certain compatibility parameters (speed, for example) that cannot be configured on the member ports.

In the ACI model, port-channels are configured as logical entities identified by a name to represent a collection of policies that can be assigned to set of ports in one or more leaf nodes. Such assignment creates one port-channel interface in each of the leaf nodes identified by an auto-generated number in the range 1 to 4096 within the leaf node, which may be same or different among the nodes for the same port-channel name. The membership of these port-channels may be same or different as well. When a port-channel is created on the FEX ports, the same port-channel name can be used to create one port-channel interface in each of the FEX devices attached to the leaf node. Thus, it is possible to create up to N+1 unique port-channel interfaces (identified by the auto-generated port-channel numbers) for each leaf node attached to N FEX modules. This is illustrated with the examples below. Port-channels on the FEX ports are identified by specifying the *fex-id* along with the port-channel name (**interface port-channel foo fex 101**, for example).



- N+1 instances per leaf of port-channel foo are possible when each leaf is connected to N FEX nodes.
- Leaf ports and FEX ports cannot be part of the same port-channel instance.
- Each FEX node can have only one instance of port-channel foo.

SUMMARY STEPS

1. **configure**
2. **template port-channel** *channel-name*
3. **[no] switchport access vlan** *vlan-id* **tenant** *tenant-name* **application** *application-name* **epg** *epg-name*
4. **channel-mode active**
5. **exit**
6. **leaf** *node-id*
7. **interface** *type*
8. **[no] channel-group** *channel-name*
9. (Optional) **lacp port-priority** *priority*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apicl# configure</code>	Enters global configuration mode.
Step 2	template port-channel <i>channel-name</i> Example: <code>apicl(config)# template port-channel foo</code>	Creates a new port-channel or configures an existing port-channel (global configuration).
Step 3	[no] switchport access vlan <i>vlan-id</i> tenant <i>tenant-name</i> application <i>application-name</i> epg <i>epg-name</i> Example: <code>apicl(config-po-ch-if)# switchport access vlan 4 tenant ExampleCorp application Web epg webEpg</code>	Deploys the EPG with the VLAN on all ports with which the port-channel is associated.
Step 4	channel-mode active Example: <code>apicl(config-po-ch-if)# channel-mode active</code> Note To enable symmetric hashing, enter the lacp symmetric-hash command: <code>apicl(config-po-ch-if)# lacp symmetric-hash</code>	Note The channel-mode command is equivalent to the mode option in the channel-group command in NX-OS. In ACI, however, this is supported for the port-channel (not on a member port). 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

	Command or Action	Purpose
Step 5	exit Example: <code>apic1(config-po-ch-if) # exit</code>	Returns to configure mode.
Step 6	leaf <i>node-id</i> Example: <code>apic1(config) # leaf 101</code>	Specifies the leaf switches to be configured. The <i>node-id</i> can be a single node ID or a range of IDs, in the form <i>node-id1-node-id2</i> , to which the configuration will be applied.
Step 7	interface <i>type</i> Example: <code>apic1(config-leaf) # interface ethernet 1/1-2</code>	Specifies the interface or range of interfaces that you are configuring to the port-channel.
Step 8	[no] channel-group <i>channel-name</i> Example: <code>apic1(config-leaf-if) # channel-group foo</code>	Assigns the interface or range of interfaces to the port-channel. Use the keyword no to remove the interface from the port-channel. To change the port-channel assignment on an interface, you can enter the channel-group command without first removing the interface from the previous port-channel.
Step 9	(Optional) lACP port-priority <i>priority</i> Example: <code>apic1(config-leaf-if) # lacp port-priority 1000</code> <code>apic1(config-leaf-if) # lacp rate fast</code>	This setting and other per-port LACP properties can be applied to member ports of a port-channel at this point. Note In the ACI model, these commands are allowed only after the ports are member of a port channel. If a port is removed from a port channel, configuration of these per-port properties are removed as well.

The following table shows various commands for global configurations of port channel properties in the ACI model. These commands can also be used for configuring overrides for port channels in a specific leaf in the (config-leaf-if) CLI mode. The configuration made on the port-channel is applied to all member ports.

CLI Syntax	Feature
[no] speed <speedValue>	Set the speed for port-channel
[no] link debounce time <time>	Set Link Debounce for port-channel
[no] negotiate auto	Configure Negotiate for port-channel
[no] cdp enable	Disable/Enable CDP for port-channel
[no] mcp enable	Disable/Enable MCP for port-channel
[no] lldp transmit	Set the transmit for port-channel
[no] lldp receive	Set the lldp receive for port-channel

CLI Syntax	Feature
spanning-tree <bpduguard bpdupfilter> <enable disable>	Configure spanning tree BPDU
[no] storm-control level <percentage> [burst-rate <percentage>]	Storm-control configuration (percentage)
[no] storm-control pps <packet-per-second> burst-rate <packets-per-second>	Storm-control configuration (packets-per-second)
[no] channel-mode { active passive on mac-pinning }	LACP mode for the link in port-channel 1
[no] lacp min-links <value>	Set minimum number of links
[no] lacp max-links <value>	Set maximum number of links
[no] lacp fast-select-hot-standby	LACP fast select for hot standby ports
[no] lacp graceful-convergence	LACP graceful convergence
[no] lacp load-defer	LACP load defer member ports
[no] lacp suspend-individual	LACP individual Port suspension
[no] lacp port-priority	LACP port priority
[no] lacp rate	LACP rate

Examples

Configure a port channel (global configuration). A logical entity foo is created that represents a collection of policies with two configurations: speed and channel mode. More properties can be configured as required.



Note

The channel mode command is equivalent to the mode option in the channel group command in NX-OS. In ACI, however, this supported for the port-channel (not on member port).

```
apic1(config)# template port-channel foo
apic1(config-po-ch-if)# switchport access vlan 4 tenant ExampleCorp application Web epg webEpg
apic1(config-po-ch-if)# speed 10G
apic1(config-po-ch-if)# channel-mode active
```

Configure ports to a port-channel in a FEX. In this example, port channel foo is assigned to ports Ethernet 1/1-2 in FEX 101 attached to leaf node 102 to create an instance of port channel foo. The leaf node will auto-generate a number, say 1002 to identify the port channel in the switch. This port channel number would be unique to the leaf node 102 regardless of how many instance of port channel foo are created.



Note The configuration to attach the FEX module to the leaf node must be done before creating port channels using FEX ports.

```
apic1(config)# leaf 102
apic1(config-leaf)# interface ethernet 101/1/1-2
apic1(config-leaf-if)# channel-group foo
```

In Leaf 102, this port channel interface can be referred to as interface port-channel foo FEX 101.

```
apic1(config)# leaf 102
apic1(config-leaf)# interface port-channel foo fex 101
apic1(config-leaf)# shut
```

Configure ports to a port channel in multiple leaf nodes. In this example, port channel foo is assigned to ports Ethernet 1/1-2 in each of the leaf nodes 101-103. The leaf nodes will auto generate a number unique in each node (which may be same or different among nodes) to represent the port-channel interfaces.

```
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/1-2
apic1(config-leaf-if)# channel-group foo
```

Add members to port channels. This example would add two members eth1/3-4 to the port-channel in each leaf node, so that port-channel foo in each node would have members eth 1/1-4.

```
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group foo
```

Remove members from port channels. This example would remove two members eth1/2, eth1/4 from the port channel foo in each leaf node, so that port channel foo in each node would have members eth 1/1, eth1/3.

```
apic1(config)# leaf 101-103
apic1(config-leaf)# interface eth 1/2,1/4
apic1(config-leaf-if)# no channel-group foo
```

Configure port-channel with different members in multiple leaf nodes. This example shows how to use the same port-channel foo policies to create a port-channel interface in multiple leaf nodes with different member ports in each leaf. The port-channel numbers in the leaf nodes may be same or different for the same port-channel foo. In the CLI, however, the configuration will be referred as interface port-channel foo. If the port-channel is configured for the FEX ports, it would be referred to as interface port-channel foo fex <fex-id>.

```
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/1-2
apic1(config-leaf-if)# channel-group foo
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 102
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group foo
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

```

apic1(config)# leaf 103
apic1(config-leaf)# interface ethernet 1/5-8
apic1(config-leaf-if)# channel-group foo
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 101/1/1-2
apic1(config-leaf-if)# channel-group foo

```

Configure per port properties for LACP. This example shows how to configure member ports of a port-channel for per-port properties for LACP.



Note

In ACI model, these commands are allowed only after the ports are member of a port channel. If a port is removed from a port channel, configuration of these per-port properties would be removed as well.

```

apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/1-2
apic1(config-leaf-if)# channel-group foo
apic1(config-leaf-if)# lacp port-priority 1000
apic1(config-leaf-if)# lacp rate fast

```

Configure admin state for port channels. In this example, a port-channel foo is configured in each of the leaf nodes 101-103 using the channel-group command. The admin state of port-channel(s) can be configured in each leaf using the port-channel interface. In ACI model, the admin state of the port-channel cannot be configured in the global scope.

```

// create port-channel foo in each leaf
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group foo

// configure admin state in specific leaf
apic1(config)# leaf 101
apic1(config-leaf)# interface port-channel foo
apic1(config-leaf-if)# shut

```

Override config is very helpful to assign specific vlan-domain, for example, to the port-channel interfaces in each leaf while sharing other properties.

```

// configure a port channel global config
apic1(config)# interface port-channel foo
apic1(config-if)# speed 1G
apic1(config-if)# channel-mode active

// create port-channel foo in each leaf
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/1-2
apic1(config-leaf-if)# channel-group foo

// override port-channel foo in leaf 102
apic1(config)# leaf 102
apic1(config-leaf)# interface port-channel foo
apic1(config-leaf-if)# speed 10G
apic1(config-leaf-if)# channel-mode on
apic1(config-leaf-if)# vlan-domain dom-foo

```


This example shows how to change port channel assignment for ports using the channel-group command. There is no need to remove port channel membership before assigning to other port channel.

```
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group foo
apic1(config-leaf-if)# channel-group bar
```

Configuring Virtual Port Channels in Leaf Nodes and FEX Devices Using the NX-OS CLI

A Virtual Port Channel (VPC) is an enhancement to port-channels that allows connection of a host or switch to two upstream leaf nodes to improve bandwidth utilization and availability. In NX-OS, VPC configuration is done in each of the two upstream switches and configuration is synchronized using peer link between the 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” 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.

The ACI model does not require a peer link and VPC configuration can be done globally for both the upstream leaf nodes. A global configuration mode called **vpc context** is introduced in ACI and VPC interfaces are represented using a type **interface vpc** that allows global configuration applicable to both leaf nodes.

Two different topologies are supported for VPC in the ACI model: VPC using leaf ports and VPC over FEX ports. It is possible to create many VPC interfaces between a pair of leaf nodes and similarly, many VPC interfaces can be created between a pair of FEX modules attached to the leaf node pairs in a straight-through topology.

VPC considerations include:

- The VPC name used is unique between leaf node pairs. For example, only one VPC 'corp' can be created per leaf pair (with or without FEX).
- Leaf ports and FEX ports cannot be part of the same VPC.
- Each FEX module can be part of only one instance of VPC corp.
- VPC context allows configuration
- The VPC context mode allows configuration of all VPCs for a given leaf pair. For VPC over FEX, the *fex-id* pairs must be specified either for the VPC context or along with the VPC interface, as shown in the following two alternative examples.

```
(config)# vpc context leaf 101 102
(config-vpc)# interface vpc Reg fex 101 101
```

or

```
(config)# vpc context leaf 101 102 fex 101 101
(config-vpc)# interface vpc Reg
```

In the ACI model, VPC configuration is done in the following steps (as shown in the examples below).



Note A VLAN domain is required with a VLAN range. It must be associated with the port-channel template.

1. VLAN domain configuration (global config) with VLAN range
2. VPC domain configuration (global config)
3. Port-channel template configuration (global config)
4. Associate the port-channel template with the VLAN domain
5. Port-channel configuration for VPC (global config)
6. Configure ports to VPC in leaf nodes
7. Configure L2, L3 for VPC in the vpc context

SUMMARY STEPS

1. **configure**
2. **vlan-domain** *name* [**dynamic**] [**type** *domain-type*]
3. **vlan** *range*
4. **vpc domain explicit** *domain-id* **leaf** *node-id1 node-id2*
5. **peer-dead-interval** *interval*
6. **exit**
7. **template port-channel** *channel-name*
8. **vlan-domain member** *vlan-domain-name*
9. **switchport access vlan** *vlan-id* **tenant** *tenant-name* **application** *application-name* **epg** *epg-name*
10. **channel-mode** **active**
11. **exit**
12. **leaf** *node-id1 node-id2*
13. **interface** *typeleaf/interface-range*
14. [**no**] **channel-group** *channel-name* **vpc**
15. **exit**
16. **exit**
17. **vpc context** **leaf** *node-id1 node-id2*
18. **interface** **vpc** *channel-name*
19. (Optional) [**no**] **shutdown**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	vlan-domain <i>name</i> [dynamic] [type <i>domain-type</i>] Example: <code>apic1(config)# vlan-domain dom1 dynamic</code>	Configures a VLAN domain for the virtual port-channel (here with a port-channel template).
Step 3	vlanrange Example: <code>apic1(config-vlan)# vlan 1000-1999</code> <code>apic1(config-vlan)# exit</code>	Configures a VLAN range for the VLAN domain and exits the configuration mode. The range can be a single VLAN or a range of VLANs.
Step 4	vpc domain explicit <i>domain-id</i> leaf <i>node-id1</i> <i>node-id2</i> Example: <code>apic1(config)# vpc domain explicit 1 leaf 101 102</code>	Configures a VPC domain between a pair of leaf nodes. You can specify the VPC domain ID in the explicit mode along with the leaf node pairs. Alternative commands to configure a VPC domain are as follows: • vpc domain [consecutive reciprocal] The consecutive and reciprocal options allow auto configuration of a VPC domain across all leaf nodes in the ACI fabric. • vpc domain consecutive <i>domain-start</i> leaf <i>start-node</i> <i>end-node</i> This command configures a VPC domain consecutively for a selected set of leaf node pairs.
Step 5	peer-dead-interval <i>interval</i> Example: <code>apic1(config-vpc)# peer-dead-interval 10</code>	Configures the time delay the Leaf switch waits to restore the vPC before receiving a response from the peer. If it does not receive a response from the peer within this time, the Leaf switch considers the peer dead and brings up the vPC with the role as a master. If it does receive a response from the peer it restores the vPC at that point. The range is from 5 seconds to 600 seconds. The default is 200 seconds.
Step 6	exit Example: <code>apic1(config-vpc)# exit</code>	Returns to global configuration mode.
Step 7	template port-channel <i>channel-name</i> Example:	Creates a new port-channel or configures an existing port-channel (global configuration).

	Command or Action	Purpose
	<code>apic1(config)# template port-channel corp</code>	All VPCs are configured as port-channels in each leaf pair. The same port-channel name must be used in a leaf pair for the same VPC. This port-channel can be used to create a VPC among one or more pairs of leaf nodes. Each leaf node will have only one instance of this VPC.
Step 8	vlan-domain member <i>vlan-domain-name</i> Example: <code>vlan-domain member dom1</code>	Associates the port channel template with the previously configured VLAN domain.
Step 9	switchport access vlan <i>vlan-id</i> tenant <i>tenant-name</i> application <i>application-name</i> epg <i>epg-name</i> Example: <code>apic1(config-po-ch-if)# switchport access vlan 4</code> <code>tenant ExampleCorp application Web epg webEpg</code>	Deploys the EPG with the VLAN on all ports with which the port-channel is associated.
Step 10	channel-mode active Example: <code>apic1(config-po-ch-if)# channel-mode active</code>	Note A port-channel must be in active channel-mode for a VPC.
Step 11	exit Example: <code>apic1(config-po-ch-if)# exit</code>	Returns to configure mode.
Step 12	leaf <i>node-id1 node-id2</i> Example: <code>apic1(config)# leaf 101-102</code>	Specifies the pair of leaf switches to be configured.
Step 13	interface <i>typeleaf/interface-range</i> Example: <code>apic1(config-leaf)# interface ethernet 1/3-4</code>	Specifies the interface or range of interfaces that you are configuring to the port-channel.
Step 14	[no] channel-group <i>channel-name vpc</i> Example: <code>apic1(config-leaf-if)# channel-group corp vpc</code>	Assigns the interface or range of interfaces to the port-channel. Use the keyword no to remove the interface from the port-channel. To change the port-channel assignment on an interface, you can enter the channel-group command without first removing the interface from the previous port-channel. Note The vpc keyword in this command makes the port-channel a VPC. If the VPC does not already exist, a VPC ID is automatically generated and is applied to all member leaf nodes.
Step 15	exit	

	Command or Action	Purpose
	Example: <code>apic1(config-leaf-if)# exit</code>	
Step 16	exit Example: <code>apic1(config-leaf)# exit</code>	
Step 17	vpc context leaf <i>node-id1 node-id2</i> Example: <code>apic1(config)# vpc context leaf 101 102</code>	The vpc context mode allows configuration of VPC to be applied to both leaf node pairs.
Step 18	interface vpc <i>channel-name</i> Example: <code>apic1(config-vpc)# interface vpc blue fex 102 102</code>	
Step 19	(Optional) [no] shutdown Example: <code>apic1(config-vpc-if)# no shut</code>	Administrative state configuration in the vpc context allows changing the admin state of a VPC with one command for both leaf nodes.

Example

This example shows how to configure a basic VPC.

```
apic1# configure
apic1(config)# vlan-domain dom1 dynamic
apic1(config-vlan)# vlan 1000-1999
apic1(config-vlan)# exit
apic1(config)# vpc domain explicit 1 leaf 101 102
apic1(config-vpc)# peer-dead-interval 10
apic1(config-vpc)# exit
apic1(config)# template port-channel corp
apic1(config-po-ch-if)# vlan-domain member dom1
apic1(config-po-ch-if)# channel-mode active
apic1(config-po-ch-if)# exit
apic1(config)# leaf 101-102
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group corp vpc
apic1(config-leaf-if)# exit
apic1(config)# vpc context leaf 101 102
```

This example shows how to configure VPCs with FEX ports.

```
apic1(config-leaf)# interface ethernet 101/1/1-2
apic1(config-leaf-if)# channel-group Reg vpc
apic1(config)# vpc context leaf 101 102
apic1(config-vpc)# interface vpc corp
apic1(config-vpc-if)# exit
apic1(config-vpc)# interface vpc red fex 101 101
```

```

apicl(config-vpc-if) # switchport
apicl(config-vpc-if) # exit
apicl(config-vpc) # interface vpc blue fex 102 102
apicl(config-vpc-if) # shut

```

Configuring FEX Connections Using Profiles with the NX-OS Style CLI

Use this procedure to configure FEX connections to leaf nodes using the NX-OS style CLI.



Note

Configuring FEX connections with FEX IDs 165 to 199 is not supported in the APIC GUI. To use one of these FEX IDs, configure the profile using the following commands.

SUMMARY STEPS

1. **configure**
2. **leaf-interface-profile** *name*
3. **leaf-interface-group** *name*
4. **fex associate** *fex-id* [**template** *template-type**fex-template-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	leaf-interface-profile <i>name</i> Example: apicl(config)# leaf-interface-profile fexIntProf1	Specifies the leaf interface profile to be configured.
Step 3	leaf-interface-group <i>name</i> Example: apicl(config-leaf-if-profile)# leaf-interface-group leafIntGrp1	Specifies the interface group to be configured.
Step 4	fex associate <i>fex-id</i> [template <i>template-type</i> <i>fex-template-name</i>] Example: apicl(config-leaf-if-group)# fex associate 101	Attaches a FEX module to a leaf node. Use the optional template keyword to specify a template to be used. If it does not exist, the system creates a template with the name and type you specified.

Example

This merged example configures a leaf interface profile for FEX connections with ID 101.

```
apic1# configure
apic1(config)# leaf-interface-profile fexIntProf1
apic1(config-leaf-if-profile)# leaf-interface-group leafIntGrp1
apic1(config-leaf-if-group)# fex associate 101
```

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 NX-OS CLI

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. In the NX-OS CLI, you can use a template to enable reflective relay on multiple ports or you can enable it on individual ports.

Before you begin

This procedure assumes that you have set up the Cisco Application Centric Infrastructure (ACI) fabric and installed the physical switches.

Enable reflective relay on one or multiple ports:

Example:

This example enables reflective relay on a single port:

```
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/2
apic1(config-leaf-if)# switchport vepa enabled
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

Example:

This example enables reflective relay on multiple ports using a template:

```
apic1(config)# template policy-group grp1
apic1(config-pol-grp-if)# switchport vepa enabled
apic1(config-pol-grp-if)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/2-4
apic1(config-leaf-if)# policy-group grp1
```

Example:

This example enables reflective relay on a port channel:

```
apic1(config)# leaf 101
apic1(config-leaf)# interface port-channel po2
apic1(config-leaf-if)# switchport vepa enabled
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)#
```

Example:

This example enables reflective relay on multiple port channels:

```
apic1(config)# template port-channel po1
apic1(config-if)# switchport vepa enabled
apic1(config-if)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group po1
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

Example:

This example enables reflective relay on a virtual port channel:

```
apic1(config)# vpc domain explicit 1 leaf 101 102
apic1(config-vpc)# exit
apic1(config)# template port-channel po4
apic1(config-if)# exit
apic1(config)# leaf 101-102
apic1(config-leaf)# interface eth 1/11-12
apic1(config-leaf-if)# channel-group po4 vpc
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# vpc context leaf 101 102
```



```
apic1(config-vpc)# interface vpc po4
apic1(config-vpc-if)# switchport vepa enabled
```

Configuring Policy Groups for Interfaces

In data center networks, typically configuration of many interfaces is the same across multiple nodes. This can be achieved in the ACI Policy Model by creating policy-groups to be shared by groups of interfaces across multiple leaf nodes. The policy-group is identified by a name similar to the port-channel; however, in case of port-channel the policies shared with the group of ports create one logical interface in each leaf while in case of a policy-group, each of the ports sharing the policies are individual physical interfaces. The policy-group concept is very similar to a port-profile in NX-OS.

SUMMARY STEPS

1. **configure**
2. **template policy-group** *policy-group-name*
3. **[no] switchport access vlan** *vlan-id* **tenant** *tenant-name* **application** *application-name* **epg** *epg-name*
4. (Apply configuration commands)
5. **exit**
6. **leaf** *node-id*
7. **interface** *type*
8. **[no] policy-group** *policy-group-name* **[force]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	template policy-group <i>policy-group-name</i> Example: apic1(config)# template policy-group pg1	Creates a new policy group or edits an existing policy group.
Step 3	[no] switchport access vlan <i>vlan-id</i> tenant <i>tenant-name</i> application <i>application-name</i> epg <i>epg-name</i> Example: apic1(config-pol-grp-if)# switchport access vlan 4 tenant ExampleCorp application Web epg webEpg	
Step 4	(Apply configuration commands) Example:	The table at the end of these steps shows various commands for configurations of policy-group for interfaces.

	Command or Action	Purpose
	<pre>apicl(config-pol-grp-if) # speed 10G apicl(config-pol-grp-if) # cdp enable</pre>	
Step 5	exit Example: <pre>apicl(config-pol-grp-if) # exit</pre>	Returns to configure mode.
Step 6	leaf node-id Example: <pre>apicl(config) # leaf 101-103</pre>	Specifies the leaf or leaves to be configured. The <i>node-id</i> can be a single node ID or a range of IDs, in the form <i>node-id1-node-id2</i> , to which the configuration will be applied.
Step 7	interface type Example: <pre>apicl(config-leaf) # interface ethernet 1/1-24</pre>	Specifies the interface or range of interfaces to which you will apply the policy group.
Step 8	[no] policy-group policy-group-name [force] Example: <pre>apicl(config-leaf-if) # policy-group pg1</pre>	Applies the policy-group to the interface or range of interfaces. Use the keyword no to remove the policy-group from the interface. Use the keyword force to delete any override configurations on the interfaces. If the specified policy-group was not configured prior to this command, this command would not implicitly create the policy-group. However, the policy-group would take effect on the interface after the policy-group has been configured in the global scope. To change the policy-group assignment on an interface, you can enter the policy-group command without first removing the previous policy-group from the interface. Note If you apply a policy-group to an interface and then assign the interface to a port-channel, the interface will lose the policy-group configuration and the policies in the port-channel will be applied.

The following table shows various commands for configurations of policy-group for interfaces.

CLI Syntax	Feature
[no] speed <speedValue>	Set the speed for Physical Interface
[no] link debounce time <time>	Set link debounce for Physical Interface
[no] negotiate auto	Configure Negotiate for Physical Interface
[no] cdp enable	Disable/Enable CDP for Physical Interface
[no] mcp enable	Disable/Enable MCP for Physical Interface

CLI Syntax	Feature
[no] lldp transmit	Set the LLDP transmit for Physical Interface
[no] lldp receive	Set the LLDP receive for Physical Interface
spanning-tree <bpduguard bpdufilter> <enable disable>	Configure spanning tree BPDU
[no] storm-control level <percentage> [burst-rate <percentage>]	Storm-control configuration (percentage)
[no] storm-control pps <packet-per-second> burst-rate <packets-per-second>	Storm-control configuration (packets-per-second)

Example

This example shows how to configure a policy-group and apply it to a range of ports in each of the leaf nodes 101-103. Each of the ports sharing the policy-group in each leaf will have the same configuration as defined in the policy-group pg1.

```

apic1# configure
apic1(config)# template policy-group pg1
apic1(config-pol-grp-if)# switchport access vlan 4 tenant ExampleCorp application Web epg
webEpg
apic1(config-pol-grp-if)# speed 10G
apic1(config-pol-grp-if)# cdp enable
apic1(config-pol-grp-if)# exit
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/1-24
apic1(config-leaf-if)# policy-group pg1

```

Configuring Overrides for Interfaces

When policy-groups are used with large number of interfaces, it may be useful to have the option to configure a set of ports for specific properties that will override the configuration in the assigned policy-group. Override configuration is allowed only if the port is assigned to a policy-group. Override configuration is not allowed for member ports of a port-channel. When a port is added to a port-channel, the override configuration is automatically removed. However, during policy-group assignment to a port that has overrides configured, the override configuration is not removed automatically and the user can decide to remove the override configuration with the force option, if required, in the policy-group command.

When a policy-group assignment is removed from a port, the override config, if exists, does not change. Similarly, the override config does not change if the port is assigned to a different policy-group (without the force option). The override config takes effect once configured and it is not removed even if the user assigns default values to all the properties in the override. To remove the override config, the user can reapply the policy-group assignment with force option. The force option, however, is not displayed in the show running-config as it is used to just remove the override config in the ACI model.

In the ACI model, overrides can be configured for a policy which may contain one or more properties. If a policy has more than one property, it is not possible to override only one property within a policy. In the CLI

framework, when the user intends to override a property for which the corresponding policy has more than one property, all other properties in the policy except the override property would be implicitly copied to the override configuration to avoid ambiguity. Such implicit copy of configuration would be reflected in the output of `show running-config` regardless of the value (including default values). Also, the copy is done only once during the configuration of the override policy and any subsequent change to the policy-group for any of the properties in that policy would have no effect on the port(s) on which the override is configured.

If the policy-group assigned to a port is not configured when the override is created, the implicit copy of properties noted above is not possible; instead, default values are assigned to properties in the override config for which the corresponding policy has more than one property. These properties shall not change for the override config when the policy-group is configured afterwards. It is recommended that user create overrides after configuring the policy-group itself or the user may need to configure the overrides in addition to the config in policy-group to get desired configuration if the config for properties in override are set to default implicitly before the configuration of the policy-group with non-default values for those properties.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **interface** *type*
4. **policy-group** *policy-group-name* **force**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apicl# configure</code>	Enters global configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apicl(config)# leaf 102</code>	Specifies the leaf or leaves to be configured. The <i>node-id</i> can be a single node ID or a range of IDs, in the form <i>node-id1-node-id2</i> , to which the configuration will be applied.
Step 3	interface <i>type</i> Example: <code>apicl(config-leaf)# interface ethernet 1/2</code>	Specifies the interface or range of interfaces with an override configuration.
Step 4	policy-group <i>policy-group-name</i> force Example: <code>apicl(config-leaf-if)# policy-group pg1 force</code>	Forces the policy-group to the interface or range of interfaces, deleting any override configurations on the interfaces.

Examples

This example shows how to apply a policy-group and then override the speed configuration for port eth1/1 in leaf node 101. In the ACI model, speed is part of a policy that also contains properties autoneg and link debounce time. As a result, those properties are copied from the speed policy-group when the override of pg1 is configured.

```

apic1# configure
apic1(config)# interface policy-group pg1
apic1(config-pol-grp-if)# speed 10G
apic1(config-pol-grp-if)# cdp enable
apic1(config-pol-grp-if)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/1-2
apic1(config-leaf-if)# policy-group pg1
apic1(config-pol-grp-if)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/1
apic1(config-leaf-if)# speed 1G
apic1(config-leaf-if)# show running-config

leaf 101
    interface ethernet 1/1
        policy-group pg1
        speed 1G
        autoneg on
        link debounce time 100

    interface ethernet 1/2
        policy-group pg1

```

This example shows how to remove the override configuration from port eth1/1 in leaf node 101.

```

apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/1
apic1(config-leaf-if)# policy-group pg1 force
apic1(config-leaf-if)# show running-config

leaf 101
    interface ethernet 1/1
        policy-group pg1

```

About Forwarding Error Correction

Forwarding Error Correction (FEC) is a method of obtaining error control in data transmission over an unreliable or noisy channel in which the source (transmitter) encodes the data in a redundant way using Error Correcting Code, and the destination (receiver) recognizes it and corrects the errors without requiring a retransmission. The available options are as follows:

- CL74-FC-FEC—Supports 25 Gbps speed.
- CL91-RS-FEC—Supports 25 and 100 Gbps speeds.
- Disable-FEC—Disables FEC.
- Inherit—The switch uses FEC based on the port transceiver type. All copper (CR4) transceivers have FC-FEC enabled on 25G. All interfaces with 100G transceivers have RS-FEC enabled.

The default is "Inherit".



Note FEC is only configurable on the front port and not on fabric ports.

Configuring FEC Using NX-OS Style CLI

SUMMARY STEPS

1. Enter the configure mode.
2. Enter the switch mode.
3. Specify the interface and port.
4. Configure FEC.
5. Exit the interface mode.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Enter the configure mode. Example: <code>apicl# configure</code>	Enters the configuration mode.
Step 2	Enter the switch mode. Example: <code>apicl(config)# leaf 104</code>	Enters the switch mode.
Step 3	Specify the interface and port. Example: <code>apicl(config-leaf)# int eth 1/4</code>	Specifies the interface and port.
Step 4	Configure FEC. Example: <code>apicl(config-leaf-if)# forward-error-correction cl91-rs-fec</code>	Configures RS-FEC. Note The default forward-error-correction value is inherit.
Step 5	Exit the interface mode. Example: <code>apicl(config-leaf-if)# exit</code>	Exits the interface mode.



CHAPTER 3

Cisco ACI Smart Licensing

This chapter contains the following sections:

- [About Smart Licensing, on page 35](#)

About Smart Licensing

Starting with Cisco Application Programming Infrastructure Controller (APIC) release 3.2(1), Smart Licensing is enabled in the Cisco ACI fabric and by extension in the Cisco APIC as a Cisco Smart Licensing-enabled product. Cisco Smart Licensing is a unified license management system that manages all the software licenses across Cisco products.

Smart Licensing has the following advantages over a traditional license:

- For the purposes of Smart Licensing, **APIC** is occasionally referred to as the **ACI controller product**.
- CSSM (Cisco Smart Software Manager) provides a central portal view to customers. Customers can view all the licenses they purchased and license usage and status. It helps prevent occurrences of license violations, expiry of subscription-based licenses, and out-of-compliance licenses.
- To support Smart Licensing, the standard CLI commands and GUI view is implemented across different Cisco products. This provides customers with a consistent user experience.
- Smart Licensing reduces the complexity of license management and makes it easier for customers to troubleshoot license-related issues.

The following URLs provide you with additional information about Smart Licensing:

- The customer log in URL to access your CSSM account: <https://software.cisco.com/>
- Cisco Smart Accounts URL: <https://www.cisco.com/c/en/us/products/software/smart-accounts.html>

The following URLs are additional resources that you can refer:

- [Training Materials and Resources](#)
- [Smart Accounts and Smart Licensing](#)

Smart Licensing Usage Guidelines and Limitations

Follow these Smart Licensing guidelines and limitations:

- The Evaluation Period countdown time is stored in the APIC. The countdown time remains intact during a software downgrade. Therefore, if the customer upgrades their APIC software version 3.2 or later once again after a downgrade, the countdown time will continue from the previous value before the downgrade. The countdown time cannot be reset. After 90 days, if no action is taken to register, the license status will display Evaluation Expired.
- If there is a license violation for a feature that is enabled on APIC, the feature functionality will not be disabled, and there will be no impact on system functionality. The system will continue to operate, but relevant faults will be raised to warn the user. The most severe fault that will be raised is **major**.
- If the registration fails, click the **Faults** tab in the APIC GUI **System > Smart Licensing** area. To see details about a specific failure, double click the listed fault.
- The DLC tool is not supported when you use the Smart Software Manager Satellite transport setting.

Pre-Registration Verifications

Verification Checklist for CSSM Configurations

The following is a user checklist for readiness and configurations required with CSSM.

1. Verify that you have the appropriate Smart Account and Virtual Accounts created.
2. If you have purchased smart-enabled licenses from Cisco Commerce, then verify that your user-purchased licenses are populated.
3. As you begin the APIC Smart Licensing registration, work with your Cisco TAC engineer to ensure that you are ready with the appropriate CSSM items.

Verification Checklist for Smart Licensing and APIC Configurations

The following is a user checklist for readiness and configurations required with the APIC.

- Your DNS settings must be configured in APIC to resolve to <https://software.cisco.com/>.

Registering for Smart Licensing Using the CLI

Registering for Smart Licensing with Direct Connect to CSSM Using the CLI

SUMMARY STEPS

1. `configure`
2. `license smart transport-mode smart-licensing`
3. `license smart register idtoken` *id token from cssm account*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	license smart transport-mode smart-licensing Example: apic1(config)# license smart transport-mode smart-licensing	Configures the Smart Licensing mode.
Step 3	license smart register idtoken <i>id token from cssm account</i> Example: apic1(config)# license smart register idtoken <id token from cssm account>	Registers with the CSSM account using the token from the account.

Registering for Smart Licensing with Transport Gateway Using the CLI

SUMMARY STEPS

1. **configure**
2. **license smart transport-mode satellite url**
http(s)://10.0.0.0:8080/Transportgateway/services/DeviceRequestHandler
3. **license smart register idtoken** *id token from cssm account*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	license smart transport-mode satellite url <i>http(s)://10.0.0.0:8080/Transportgateway/services/DeviceRequestHandler</i> Example: apic1(config)# license smart transport-mode satellite url http(s)://<ip address hostname of transport gateway>:<http(s) port>/Transportgateway/services/DeviceRequestHandler	Configures the Transport Gateway mode and URL.
Step 3	license smart register idtoken <i>id token from cssm account</i> Example: apic1(config)# license smart register idtoken <id token from cssm account>	Registers with the CSSM using the token from the CSSM Smart account or the CSSM Virtual account.

Registering for Smart Licensing with Smart Software Manager Satellite Using the CLI

SUMMARY STEPS

1. **configure**
2. **license smart transport-mode satellite url**
http(s)://10.0.10.1:8080/Transportgateway/services/DeviceRequestHandler
3. **license smart register idtoken** *id token from smart software manager satellite*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters configuration mode.
Step 2	license smart transport-mode satellite url <i>http(s)://10.0.10.1:8080/Transportgateway/services/DeviceRequestHandler</i> Example: apicl(config)# license smart transport-mode satellite url <i>http(s)://<ip address hostname of transport gateway>:<http(s) port>/Transportgateway/services/DeviceRequestHandler</i>	Configures the Smart Software Manager Satellite mode and URL.
Step 3	license smart register idtoken <i>id token from smart software manager satellite</i> Example: apicl(config)# license smart register idtoken <i><id token from smart software manager satellite></i>	Registers with the Satellite using the token from the Smart Software Manager Satellite account. Note Note : Do not use the token from the CSSM account.

Registering for Smart Licensing with HTTP or HTTPS Proxy Using the CLI

SUMMARY STEPS

1. **configure**
2. **license smart transport-mode proxy ip-address** *ip address* **port** *port number*
3. **license smart register idtoken** *id token from cssm account*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters configuration mode.

	Command or Action	Purpose
Step 2	license smart transport-mode proxy <i>ip-address ip address</i> <i>port port number</i> Example: apic1(config)# license smart transport-mode proxy ip-address 10.0.0.248 port 4440	Configures the proxy mode, the IP address or hostname and the http(s) port.
Step 3	license smart register idtoken <i>id token from cssm account</i> Example: apic1(config)# license smart register idtoken <id token from cssm account>	Registers with the CSSM account using the token from the CSSM smart account or the CSSM virtual account.



CHAPTER 4

Configuring APIC High Availability

- [About Cold Standby for APIC Cluster, on page 41](#)
- [Switching Over Active APIC with Standby APIC Using CLI, on page 42](#)

About Cold Standby for APIC Cluster

The Cold Standby functionality for an APIC cluster enables you to operate the APICs in a cluster in an Active/Standby mode. In an APIC cluster, the designated active APICs share the load and the designated standby APICs can act as a replacement for any of the APICs in an active cluster.

As an admin user, you can set up the Cold Standby functionality when the APIC is launched for the first time. We recommend that you have at least three active APICs in a cluster, and one or more standby APICs. As an admin user, you can initiate the switch over to replace an active APIC with a standby APIC.

Important Notes

- The standby APIC is automatically updated with firmware updates to keep the backup APIC at same firmware version as the active cluster.
- During an upgrade process, once all the active APICs are upgraded, the standby APIC is also be upgraded automatically.
- Temporary IDs are assigned to standby APICs. After a standby APIC is switched over to an active APIC, a new ID is assigned.
- Admin login is not enabled on standby APIC. To troubleshoot Cold Standby, you must log in to the standby using SSH as *rescue-user*.
- During switch over the replaced active APIC is powered down, to prevent connectivity to the replaced APIC.
- Switch over fails under the following conditions:
 - If there is no connectivity to the standby APIC.
 - If the firmware version of the standby APIC is not the same as that of the active cluster.
- After switching over a standby APIC to 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 standby APIC after a fail over.

- Standby (new active) APIC may not retain its out of band address if more than 1 active APICs are down or unavailable.
- Standby (new active) APIC may not retain its out of band address if it is in a different subnet than active APIC. This limitation is only applicable for APIC release 2.x.
- Standby (new active) APIC may not retain its IPv6 out of band address. This limitation is not applicable starting from APIC release 3.1x.
- Standby (new active) APIC may not retain its out of band address if you have configured non Static OOB Management IP address policy for replacement (old active) APIC.

**Note**

In case you observe any of the limitations, in order to retain standby APICs out of band address, you must manually change the OOB policy for replaced APIC after the replace operation is completed successfully.

- We recommend keeping standby APICs in same POD as the active APICs it may replace.
- There must be three active APICs in order to add a standby APIC.
- The standby APIC does not participate in policy configuration or management.
- No information is replicated to standby controllers, including admin credentials.

Switching Over Active APIC with Standby APIC Using CLI

Use this procedure to switch over an active APIC with a standby APIC.

SUMMARY STEPS

1. **replace-controller replace** *ID number Backup serial number*
2. **replace-controller reset** *ID number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	replace-controller replace <i>ID number Backup serial number</i> Example: <pre>apicl#replace-controller replace 2 FCH1804V27L Do you want to replace APIC 2 with a backup? (Y/n): Y</pre>	Replaces an active APIC with an standby APIC.
Step 2	replace-controller reset <i>ID number</i> Example: <pre>apicl# replace-controller reset 2 Do you want to reset failover status of APIC 2? (Y/n): Y</pre>	Resets fail over status of the active controller.



CHAPTER 5

Configuring Tenants

- [Creating a Tenant, VRF, and Bridge Domain, on page 43](#)
- [Additional Bridge Domain Configuration, on page 46](#)
- [Configuring an Enforced Bridge Domain, on page 48](#)
- [Creating an Application Endpoint Group, on page 50](#)
- [Configuring Legacy Forwarding Mode in the Bridge Domain, on page 53](#)
- [Configuring Contracts, on page 55](#)
- [Contract Inheritance, on page 59](#)
- [Configuring Contract Preferred Groups, on page 68](#)
- [Exporting a Contract to Another Tenant, on page 71](#)
- [Configuring Contract or Subject Exceptions, on page 73](#)
- [Creating Quota Management, on page 75](#)

Creating a Tenant, VRF, and Bridge Domain

This topic describes the following steps in the basic provisioning of a new tenant:

1. Create a tenant
2. Associate the tenant with a security domain
3. Create a VRF for the tenant
4. Create a bridge domain for endpoint groups within the tenant

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **security domain** *domain-name*
4. **[no] vrf context** *vrf-name*
5. **[no] contract** {**provider** | **consumer**} *contract-name*
6. **exit**
7. **[no] bridge-domain** *bd-name*
8. **[no] vrf member** *vrf-name*
9. **exit**

10. **interface bridge-domain** *bd-name*
11. **[no] {ip | ipv6} address** *address/mask-length* [**scope** {**private** | **public**}] [**secondary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apic1(config)# tenant exampleCorp</code>	Creates a tenant if it does not exist and enters the tenant configuration mode.
Step 3	security domain <i>domain-name</i> Example: <code>apic1(config-tenant)# security domain exampleCorp_dom1</code>	Associates the tenant with one or more security domains.
Step 4	[no] vrf context <i>vrf-name</i> Example: <code>apic1(config-tenant)# vrf context exampleCorp_v1</code>	Creates a private network (VRF) for the tenant. A tenant can have one or more VRFs configured.
Step 5	[no] contract { provider consumer } <i>contract-name</i> Example: <code>apic1(config-tenant-vrf)# contract provider web</code>	Provide or consume contracts for all the EPGs under the VRF.
Step 6	exit Example: <code>apic1(config-tenant-vrf)# exit</code>	Returns to the tenant configuration mode.
Step 7	[no] bridge-domain <i>bd-name</i> Example: <code>apic1(config-tenant)# bridge-domain exampleCorp_b1</code>	Creates or deletes a bridge domain under the tenant. Enters bridge domain configuration mode.
Step 8	[no] vrf member <i>vrf-name</i> Example: <code>apic1(config-tenant-bd)# vrf member exampleCorp_v1</code>	Assigns the bridge-domain to a VRF.
Step 9	exit Example: <code>apic1(config-tenant-bd)# exit</code>	Returns to the tenant configuration mode.
Step 10	interface bridge-domain <i>bd-name</i> Example:	Enters tenant interface configuration mode to enable routing and to apply interfaces to the bridge domain.

	Command or Action	Purpose
	apic1(config-tenant)# interface bridge-domain exampleCorp_b1	
Step 11	[no] {ip ipv6} address <i>address/mask-length</i> [scope {private public}] [secondary] Example: <pre>apic1(config-tenant-if)# ip address 172.1.1.1/24 apic1(config-tenant-if)# ipv6 address 2001:1:1::1/64</pre>	Assigns or removes the gateway IP address of the bridge domain and enters the IP address mode to configure optional IP address properties. The scope of the gateway address can be one of the following: • Public—Can be advertised to external Layer 3 networks through routing protocols (BGP, OSPF, EIGRP). • Private—Not advertised to external Layer 3 networks. The optional secondary keyword allows you to configure a secondary gateway address.

Examples

This example shows the basic configuration of a tenant including assignment to a security domain, creation of a VRF with contracts, and creation of a bridge domain.

```
apic1# configure
apic1(config)# tenant exampleCorp
apic1(config-tenant)# security domain exampleCorp_dom1
apic1(config-tenant)# vrf context exampleCorp_v1
apic1(config-tenant-vrf)# contract enforce
apic1(config-tenant-vrf)# contract provider web
apic1(config-tenant-vrf)# contract consumer db
apic1(config-tenant-vrf)# contract provider icmp
apic1(config-tenant-vrf)# contract consumer icmp
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# bridge-domain exampleCorp_b1
apic1(config-tenant-bd)# vrf member exampleCorp_v1
apic1(config-tenant-bd)# exit
apic1(config-tenant)# interface bridge-domain exampleCorp_b1
apic1(config-tenant-interface)# ip address 172.1.1.1/24
apic1(config-tenant-interface)# ipv6 address 2001:1:1::1/64
apic1(config-tenant-interface)# exit
```

This example shows the VRF configuration specific to a leaf.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# vrf context exampleCorp_v1 tenant exampleCorp
apic1(config-leaf-vrf)# ip route 1.2.3.4 5.6.7.8
```

This example shows the VRF configuration specific to a leaf interface.

```
apic1# configure
```

```

apic1(config)# leaf 101
apic1(config-leaf)# int eth 1/1
apic1(config-leaf-if)# vrf member exampleCorp_v1 tenant exampleCorp

```

What to do next

Add an application profile, create an application endpoint group (EPG), and associate the EPG to the bridge domain.

Additional Bridge Domain Configuration

This topic describes the following configurations for a bridge domain:

- Configuring a MAC address
- Configuring a DHCP relay address
- Configuring route leaking for shared services

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **interface bridge-domain** *bd-name*
4. (Optional) **mac-address** *mac-address*
5. (Optional) **no mac-address**
6. (Optional) **[no] ip dhcp relay address** *tenant tenant-name dhcp-address {application app-name epg epg-name | external-l2 l2-epg-name | external-l3 l3-epg-name}*
7. (Optional) **[no] {ip | ipv6} shared address** *address/mask-length provider application app-name epg epg-name*
8. (Optional) **[no] {ip | ipv6} shared address** *address/mask-length consumer application any epg any*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic# configure	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: apic(config-tenant)# tenant exampleCorp	Enters the tenant configuration mode.
Step 3	interface bridge-domain <i>bd-name</i> Example: apic(config-tenant)# interface bridge-domain exampleCorp_bdl	Enters tenant interface configuration mode to configure the bridge domain.

	Command or Action	Purpose
Step 4	(Optional) mac-address <i>mac-address</i> Example: apic(config-tenant-interface)# mac-address 1234.5678.abcd	Configures the MAC address to be used in the ARP reply for the pervasive gateway functionality.
Step 5	(Optional) no mac-address Example: apic(config-tenant-interface)# no mac-address	Changes the MAC address to its default.
Step 6	(Optional) [no] ip dhcp relay address tenant <i>tenant-name</i> dhcp-address { application <i>app-name</i> epg <i>epg-name</i> external-l2 <i>l2-epg-name</i> external-l3 <i>l3-epg-name</i> } Example: apic(config-tenant-interface)# ip dhcp relay address 192.0.20.1 tenant exampleCorp application app1 epg epg1	Sets or removes a DHCP relay address for the bridge-domain along with any supported options.
Step 7	(Optional) [no] {ip ipv6} shared address address/mask-length provider application <i>app-name</i> epg <i>epg-name</i> Example: apic(config-tenant-interface)# ip shared address 7.8.9.1/24 provider application app2 epg epg2	Route leaking is allowed across VRFs to provide common services like DHCP, DNS for multiple tenant VRFs. Shared service is enabled by marking subnets as provider or consumer subnets and specifying the EPGs providing the shared service.
Step 8	(Optional) [no] {ip ipv6} shared address address/mask-length consumer application any epg any Example: apic(config-tenant-interface)# ip shared address 3.2.3.4/24 consumer application any epg any	See the previous step.

Examples

```

apic1# configure
apic1(config)# tenant exampleCorp
apic1(config-tenant)# interface bridge-domain exampleCorp_bd1
apic1(config-tenant-interface)# mac-address 1234.5678.abcd
apic1(config-tenant-interface)# ip dhcp relay address 192.0.20.1 tenant exampleCorp application
app1 epg epg1
apic1(config-tenant-interface)# ip shared address 1.2.3.4/24 provider application any
apic1(config-tenant-interface)# ip shared address 3.2.3.4/24 consumer application any epg
any
apic1(config-tenant-interface)# exit
apic1(config-tenant)# exit
apic1(config)# tenant my_dhcp_provider
apic1(config-tenant)# interface bridge-domain bd_dhcp
apic1(config-tenant-interface)# ip shared address 7.8.9.1/24 provider application app2 epg
epg2

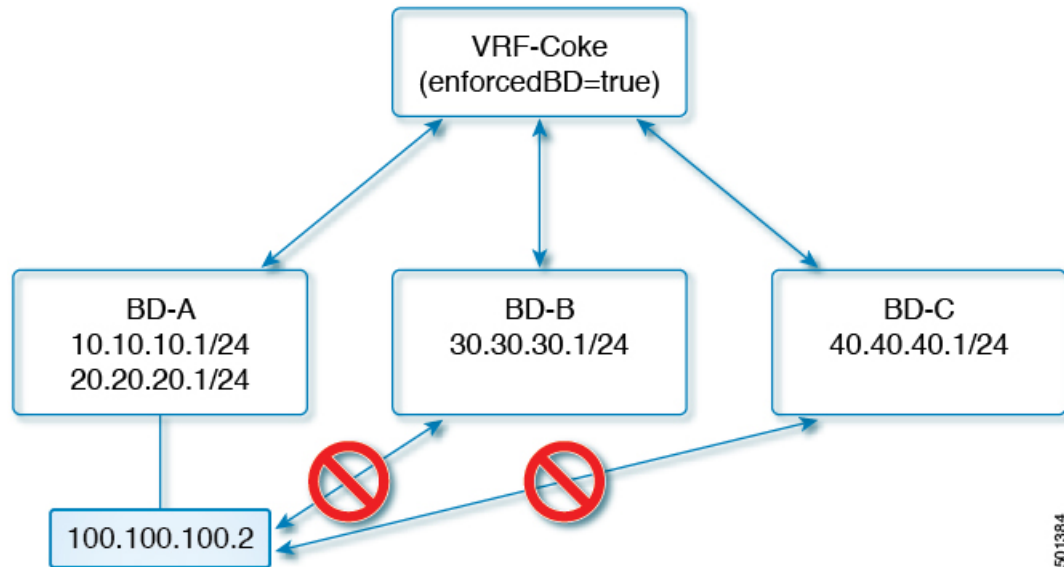
```

Configuring an Enforced Bridge Domain

An enforced bridge domain (BD) configuration entails creating an endpoint in a subject endpoint group (EPG) which can only ping subnet gateways within the associated bridge domain.

With this configuration, you can then create a global exception list of IP addresses which can ping any subnet gateway.

Figure 1: Enforced Bridge Domain



Note

- The exception IP addresses can ping all of the BD gateways across all of your VRFs.
- A loopback interface configured for an L3 out 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, the peer subnet must be added to the allowed exception subnets.

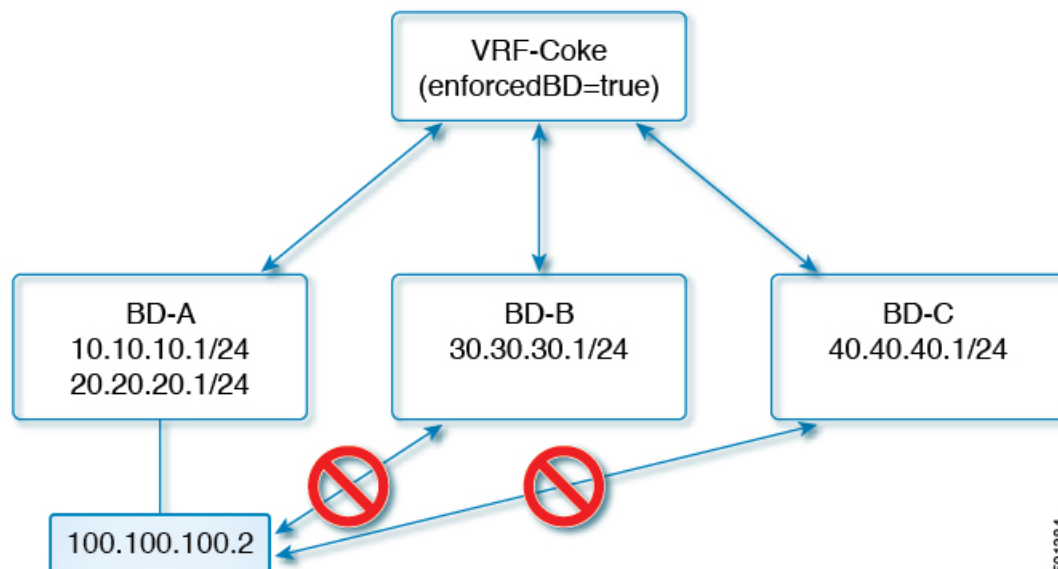
Otherwise, eBGP traffic is blocked because the source IP address exists in a different subnet than the L3out interface subnet.

Configuring an Enforced Bridge Domain

An enforced bridge domain (BD) configuration entails creating an endpoint in a subject endpoint group (EPG) which can only ping subnet gateways within the associated bridge domain.

With this configuration, you can then create a global exception list of IP addresses which can ping any subnet gateway.

Figure 2: Enforced Bridge Domain



501384

**Note**

- The exception IP addresses can ping all of the BD gateways across all of your VRFs.
 - A loopback interface configured for an L3 out 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, the peer subnet must be added 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.

Configuring an Enforced Bridge Domain Using the NX-OS Style CLI

This section provides information on how to configure your enforced bridge domain using the NX-OS style command line interface (CLI).

SUMMARY STEPS

1. Create and enable the tenant:
2. Add the subnet to the exception list.

DETAILED STEPS

Step 1 Create and enable the tenant:

Example:

In the following example ("cokeVrf") is created and enabled.

```
apic1(config-tenant)# vrf context cokeVrf
apic1(config-tenant-vrf)# bd-enforce enable
apic1(config-tenant-vrf)# exit
apic1(config-tenant)#exit
```

Step 2 Add the subnet to the exception list.

Example:

```
apic1(config)#bd-enf-exp-ip add1.2.3.4/24
apic1(config)#exit
```

You can confirm if the enforced bridge domain is operational using the following type of command:

```
apic1# show running-config all | grep bd-enf
bd-enforce enable
bd-enf-exp-ip add 1.2.3.4/24
```

Example

The following command removes the subnet from the exception list:

```
apic1(config)# no bd-enf-exp-ip 1.2.3.4/24
apic1(config)#tenant coke
apic1(config-tenant)#vrf context cokeVrf
```

What to do next

To disable the enforced bridge domain run the following command:

```
apic1(config-tenant-vrf)# no bd-enforce enable
```

Creating an Application Endpoint Group

This topic describes the following steps in the basic provisioning of a static application EPG:

1. Create an application profile within the tenant
2. Create an EPG in the application profile
3. Assign a bridge domain to the EPG
4. Deploy the EPG to a Layer 2 interface

Before you begin

Before you can create an application profile and an application endpoint group (EPG), you must create a VLAN domain, tenant, VRF, and bridge domain.

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **[no] application** *app-name*
4. **[no] epg** *epg-name*
5. **[no] bridge-domain member** *epg-name*
6. **exit**
7. **exit**
8. **exit**
9. **leaf** *node-id*
10. **interface** *type*
11. (Optional) **switchport**
12. **vlan-domain member** *domain-name*
13. **switchport trunk allowed vlan** *vlan-id* **tenant** *tenant-name* **app** *app-name* **epg** *epg-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apic1(config)# tenant exampleCorp</code>	Enters the tenant configuration mode.
Step 3	[no] application <i>app-name</i> Example: <code>apic1(config-tenant)# application OnlineStore</code>	Creates an application profile and enters application profile configuration mode.
Step 4	[no] epg <i>epg-name</i> Example: <code>apic1(config-tenant-app)# epg exampleCorp_webepg1</code>	Creates (or deletes) an EPG in the application profile and enters EPG configuration mode.
Step 5	[no] bridge-domain member <i>epg-name</i> Example: <code>apic1(config-tenant-app-epg)# bridge-domain member exampleCorp_b1</code>	Associates the EPG to the bridge domain. Every EPG must belong to a BD.
Step 6	exit Example: <code>apic1(config-tenant-app-epg)# exit</code>	Returns to the tenant application configuration mode.
Step 7	exit Example:	Returns to the tenant configuration mode.

	Command or Action	Purpose
	<code>apic1(config-tenant-app)# exit</code>	
Step 8	exit Example: <code>apic1(config-tenant)# exit</code>	Returns to the global configuration mode.
Step 9	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 10	interface <i>type</i> Example: <code>apic1(config-leaf)# interface eth 1/2</code>	Specifies the interface that you are configuring. For an Ethernet port, use “ethernet slot / port.”
Step 11	(Optional) switchport Example: <code>apic1(config-leaf-if)# switchport</code>	Because layer 2 is the default state of a port, this command is only needed when the port must be converted from a layer 3 configuration.
Step 12	vlan-domain member <i>domain-name</i> Example: <code>apic1(config-leaf-if)# vlan-domain member dom1</code>	Associates the interface with a VLAN domain.
Step 13	switchport trunk allowed vlan <i>vlan-id</i> tenant <i>tenant-name</i> app <i>app-name</i> epg <i>epg-name</i> Example: <code>apic1(config-leaf-if)# switchport trunk allowed vlan 10 tenant exampleCorp application OnlineStore epg exampleCorp_webepg1</code>	Deploys the EPG on the interface and identifies the EPG through EPG-to-VLAN mapping. This configuration applies only to static EPG deployment. If the VLAN is in use for another EPG or external SVI, you must delete the VLAN configuration before using it for this EPG. Note The interface must be associated with a VLAN domain or this command is rejected.

Examples

This example shows how to create an application EPG deployed to a layer 2 port.

```

apic1# configure
apic1(config)# tenant exampleCorp
apic1(config-tenant)# application OnlineStore
apic1(config-tenant-app)# epg exampleCorp_webepg1
apic1(config-tenant-app-epg)# bridge-domain member exampleCorp_b1
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)# exit
apic1(config-tenant)# exit

apic1(config)# leaf 101
apic1(config-leaf)# interface eth 1/2
apic1(config-leaf-if)# switchport
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk allowed vlan 10 tenant exampleCorp application

```



```
OnlineStore epg exampleCorp_webepg1
```

This example shows how to deploy the EPG to a port channel.

```
apic1(config)# leaf 101
apic1(config-leaf)# interface port-channel po1
apic1(config-leaf-if)# switchport
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk allowed vlan 10 tenant exampleCorp application
OnlineStore epg exampleCorp_webepg1
```

What to do next

Map a VLAN on a port to the EPG.

Configuring Legacy Forwarding Mode in the Bridge Domain

Legacy forwarding mode allows switching and routing without the use of contracts or EPGs. In this mode, the VLAN on a port directly maps to a bridge domain. The **legacy forwarding vlan** command automatically creates all necessary objects so that no EPG-related configuration is required.

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **bridge-domain** *bd-name*
4. **[no] legacy forwarding vlan** *vlan-id* **vlan-domain** *vlan-domain-name*
5. **exit**
6. **exit**
7. **leaf** *node-id*
8. **interface** *type*
9. **[no] switchport trunk allowed vlan** *vlan-id* **tenant** *tenant-name* **legacy-forwarding**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apic1(config)# tenant exampleCorp</code>	Enters the tenant configuration mode.
Step 3	bridge-domain <i>bd-name</i> Example:	Enters tenant interface configuration mode to configure the bridge domain.

	Command or Action	Purpose
	<code>apic1(config-tenant) # bridge-domain exampleCorp_b1</code>	
Step 4	[no] legacy forwarding vlan <i>vlan-id</i> vlan-domain <i>vlan-domain-name</i> Example: <pre>apic1(config-tenant-bd) # legacy-forwarding vlan 50 vlan-domain dom1</pre>	Maps the VLAN to the bridge domain.
Step 5	exit Example: <pre>apic1(config-tenant-bd) # exit</pre>	Returns to the tenant configuration mode.
Step 6	exit Example: <pre>apic1(config-tenant) # exit</pre>	Returns to the global configuration mode.
Step 7	leaf <i>node-id</i> Example: <pre>apic1(config) # leaf 101</pre>	Specifies the leaf to be configured.
Step 8	interface <i>type</i> Example: <pre>apic1(config-leaf) # interface eth 1/1</pre>	Specifies the interface that you are configuring. For an Ethernet port, use ethernet slot/port .
Step 9	[no] switchport trunk allowed vlan <i>vlan-id</i> tenant <i>tenant-name</i> legacy-forwarding Example: <pre>apic1(config-leaf-if) # switchport trunk allowed vlan 50 tenant exampleCorp legacy-forwarding</pre>	Enables the VLAN on the interface and associates it to the tenant bridge domain that uses the VLAN in the legacy forwarding mode.

Examples

This example shows how to configure legacy forwarding mode for forwarding between bridge domains.

```
apic1# configure
apic1(config) # tenant exampleCorp
apic1(config-tenant) # bridge-domain exampleCorp_b1
apic1(config-tenant-bd) # legacy-forwarding vlan 50 vlan-domain dom1
apic1(config-tenant-bd) # exit
apic1(config-tenant) # bridge-domain exampleCorp_b2
apic1(config-tenant-bd) # legacy-forwarding vlan 60 vlan-domain dom1
apic1(config-tenant-bd) # exit
apic1(config-tenant) # exit

apic1(config) # leaf 101
apic1(config-leaf) # interface eth 1/1
apic1(config-leaf-if) # vlan-domain member dom1
```

```

apic1(config-leaf-if)# switchport trunk allowed vlan 50 tenant exampleCorp legacy-forwarding
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface eth 1/2
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk allowed vlan 60 tenant exampleCorp legacy-forwarding

```

Configuring Contracts

Contracts are configured under a tenant with the following tasks:

- Define filters as access lists
- Define the contract and subjects
- Link the contract to an EPG

The tasks need not follow this order. For example, you can link a contract name to an EPG before you have defined the contract.



Note

Filters (ACLs) in APIC use **match** instead of **permit** | **deny** as in the traditional NX-OS ACL. The purpose of a filter entry is only to match a given traffic flow. The traffic will be permitted or denied when the ACL is applied on a contract or on a taboo contract.

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **access-list** *acl-name*
4. (Optional) **match** {arp | icmp | ip}
5. (Optional) **match** {tcp | udp} [*src from*[-to]] [*dest from*[-to]]
6. (Optional) **match raw** *options*
7. **exit**
8. **contract** *contract-name*
9. **subject** *subject-name*
10. (Optional) [**no**] **access-group** *acl-name* [in | out | both]
11. (Optional) [**no**] **label name** *label-name* {provider | consumer}
12. (Optional) [**no**] **label match** {provider | consumer} [any | one | all | none]
13. **exit**
14. **exit**
15. **application** *app-name*
16. **epg** *epg-name*
17. **bridge-domain member** *bd-name*
18. **contract provider** *provider-contract-name*
19. **contract consumer** *consumer-contract-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>tenant exampleCorp</code>	Creates a tenant if it does not exist and enters the tenant configuration mode.
Step 3	access-list <i>acl-name</i> Example: <code>apic1(config-tenant)# access-list http_acl</code>	Creates an access list (filter) that can be used in a contract.
Step 4	(Optional) match { <i>arp</i> <i>icmp</i> <i>ip</i> } Example: <code>apic1(config-tenant-acl)# match arp</code>	Creates a rule to match traffic of the selected protocol.
Step 5	(Optional) match { <i>tcp</i> <i>udp</i> } [<i>src from</i> [- <i>to</i>]] [<i>dest from</i> [- <i>to</i>]] Example: <code>apic1(config-tenant-acl)# match tcp dest 80</code> <code>apic1(config-tenant-acl)# match tcp dest 443</code>	Creates a rule to match TCP or UDP traffic.
Step 6	(Optional) match raw <i>options</i> Example: <code>apic1(config-tenant-acl)#</code>	Creates a rule to match a raw vzEntry.
Step 7	exit Example: <code>apic1(config-tenant-acl)# exit</code>	Returns to the tenant configuration mode.
Step 8	contract <i>contract-name</i> Example: <code>apic1(config-tenant)# contract web80</code>	Creates a contract and enters the contract configuration mode.
Step 9	subject <i>subject-name</i> Example: <code>apic1(config-tenant-contract)# subject web80</code>	Creates a contract subject and enters the subject configuration mode.
Step 10	(Optional) [no] access-group <i>acl-name</i> [in out both] Example: <code>apic1(config-tenant-contract-subj)# access-group http_acl both</code>	Adds (removes) an access list from the contract, specifying the direction of the traffic to be matched.

	Command or Action	Purpose
Step 11	(Optional) [no] label name <i>label-name</i> {provider consumer} Example: <code>apicl (config-tenant-contract-subj) #</code>	Adds (removes) a provider or consumer label to the subject.
Step 12	(Optional) [no] label match {provider consumer} [any one all none] Example: <code>apicl (config-tenant-contract-subj) #</code>	Specifies the match type for the provider or consumer label: • any—Match if any label is found in the contract relation. • one—Match if exactly one label is found in the contract relation. • all—Match if all labels are found in the contract relation. • none—Match if no labels are found in the contract relation.
Step 13	exit Example: <code>apicl (config-tenant-contract-subj) # exit</code>	Returns to the contract configuration mode.
Step 14	exit Example: <code>apicl (config-tenant-contract) # exit</code>	Returns to the tenant configuration mode.
Step 15	application <i>app-name</i> Example: <code>apicl (config-tenant) # application OnlineStore</code>	Enters application configuration mode.
Step 16	epg <i>epg-name</i> Example: <code>apicl (config-tenant-app) # epg exampleCorp_webepg1</code>	Enters configuration mode for the EPG to be linked to the contract.
Step 17	bridge-domain member <i>bd-name</i> Example: <code>apicl (config-tenant-app-epg) # bridge-domain member exampleCorp_bd1</code>	Specifies the bridge domain for this EPG.
Step 18	contract provider <i>provider-contract-name</i> Example: <code>apicl (config-tenant-app-epg) # contract provider web80</code>	Specifies the provider contract for this EPG. Communication with this EPG can be initiated from other EPGs as long as the communication complies with this provider contract.

	Command or Action	Purpose
Step 19	contract consumer <i>consumer-contract-name</i> Example: <pre>apic1(config-tenant-app-epg)# contract consumer rmi99</pre>	Specifies the consumer contract for this EPG. The endpoints in this EPG may initiate communication with any endpoint in an EPG that is providing this contract.

Examples

This example shows how to create and apply contracts to an EPG.

```
apic1# configure
apic1(config)# tenant exampleCorp

# CREATE FILTERS
apic1(config-tenant)# access-list http_acl
apic1(config-tenant-acl)# match tcp dest 80
apic1(config-tenant-acl)# match tcp dest 443
apic1(config-tenant-acl)# exit

# CREATE CONTRACT WITH FILTERS
apic1(config-tenant)# contract web80
apic1(config-tenant-contract)# subject web80
apic1(config-tenant-contract-subj)# access-group http_acl both
apic1(config-tenant-contract-subj)# exit
apic1(config-tenant-contract)# exit

# ASSOCIATE CONTRACTS TO EPG
apic1(config-tenant)# application OnlineStore
apic1(config-tenant-app)# epg exampleCorp_webepg1
apic1(config-tenant-app-epg)# bridge-domain member exampleCorp_bdl
apic1(config-tenant-app-epg)# contract consumer rmi99
apic1(config-tenant-app-epg)# contract provider web80
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)#exit
apic1(config-tenant)#exit

# ASSOCIATE PORT AND VLAN TO EPG
apic1(config)#leaf 101
apic1(config-leaf)# interface ethernet 1/4
apic1(config-leaf-if)# switchport trunk allowed vlan 102 tenant exampleCorp application
OnlineStore epg exampleCorp_webepg1
```

This example shows a simpler method for defining a contract by declaring the filters inline in the contract itself.

```
apic1# configure
apic1(config)# tenant exampleCorp
apic1(config-tenant)# contract web80
apic1(config-tenant-contract)# match tcp 80
apic1(config-tenant-contract)# match tcp 443
```

Contract Inheritance

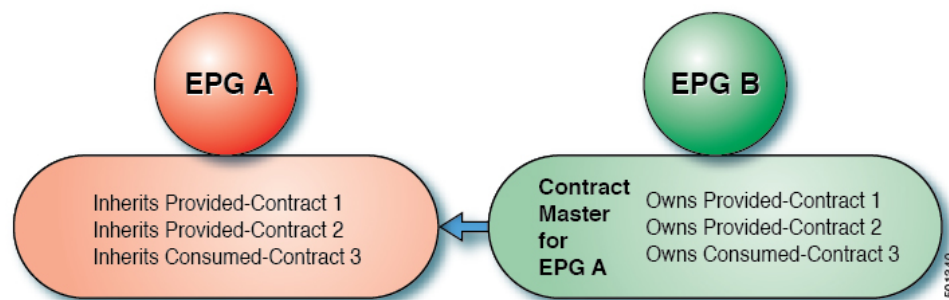
About Contract Inheritance

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 3: 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.
- Contract subject label and EPG label 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 only uses the subject label configured under EPG B and not a collection of labels from both EPGs.

- 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 or uSeg EPG Contract Inheritance Using the NX-OS Style CLI

To configure contract inheritance for application or uSeg EPGs, use the following commands:

Before you begin

Configure the tenant, application profile, and bridge-domain to be used by the EPGs.

Configure the contracts to be shared by the EPGs at the VRF level.

Procedure

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: apic1# (config) tenant Tn1	Creates or specifies the tenant to be configured; and enters into tenant configuration mode.
Step 3	application <i>application-name</i> Example: apic1(config-tenant)# application AP1	Creates or specifies an application and enters into application mode.
Step 4	epg <i>epg-name</i> [type <i>micro-segmented</i>] Example: apic1(config-tenant-app)# epg AEPg403	Creates or specifies the application or uSeg EPG to be configured and enters into EPG configuration mode. For uSeg EPGs add the type. In this example, this is the application EPG contract master.
Step 5	bridge-domain member <i>bd-name</i> Example: apic1(config-tenant-app-epg)# bridge-domain member T1BD1	Associates the EPG with the bridge domain.
Step 6	contract consumer <i>contract-name</i> Example:	Adds a contract to be consumed by this EPG.

	Command or Action	Purpose
	<code>apic1(config-tenant-app-epg)# contract consumer cctr5</code>	
Step 7	contract provider [label label] Example: <code>apic1(config-tenant-app-epg)# contract provider T1ctrl_cif</code>	Adds a contract to be provided by this EPG, including an optional list of subject or EPG labels (must be previously configured).
Step 8	exit Example: <code>apic1(config-tenant-app-epg)# exit</code>	Exits the configuration mode
Step 9	epg epg-name [type micro-segmented] Example: <code>apic1(config-tenant-app)# epg AEPg404</code>	Creates or specifies the application or uSeg EPG to be configured and enters into EPG configuration mode. For uSeg EPGs add the type. In this example, this is the EPG inheriting contracts.
Step 10	bridge-domain member bd-name Example: <code>apic1(config-tenant-app-epg)# bridge-domain member T1BD1</code>	Associates the EPG with the bridge domain.
Step 11	inherit-from-epg application application-nameepg EPG-contract-master-name] Example: <code>apic1(config-tenant-app-epg)# inherit-from-epg application AP1 epg AEPg403</code>	Configures this EPG to inherit contracts from the EPG contract master.
Step 12	exit Example: <code>apic1(config-tenant-app-epg)# exit</code>	Exits the configuration mode
Step 13	epg epg-name [type micro-segmented] Example: <code>apic1(config-tenant-app)# epg uSeg1_403_10 type micro-segmented</code>	Creates or specifies the application or uSeg EPG to be configured and enters into EPG configuration mode. In this example, this is the uSeg EPG contract master.
Step 14	bridge-domain member bd-name Example: <code>apic1(config-tenant-app-epg)# bridge-domain member T1BD1</code>	Associates the EPG with the bridge domain.
Step 15	contract provider [label label] Example: <code>apic1(config-tenant-app-epg)# contract provider T1ctrl_uSeg_l3out</code>	Adds a contract to be provided by this EPG, including an optional list of subject or EPG labels (must be previously configured).

	Command or Action	Purpose
Step 16	attribute-logical-expression <i>logical-expression</i> Example: <pre>apic1(config-tenant-app-epg)# attribute-logical-expression 'ip equals 192.168.103.10 force'</pre>	Adds a logical expression to the uSeg EPG as matching criteria.
Step 17	exit Example: <pre>apic1(config-tenant-app-epg)# exit</pre>	Exits the configuration mode
Step 18	epg <i>epg-name</i> [type <i>micro-segmented</i>] Example: <pre>apic1(config-tenant-app)# epg uSeg1_403_30 type micro-segmented</pre>	Creates or specifies the application or uSeg EPG to be configured and enters into EPG configuration mode. In this example, this is the uSeg EPG that inherits contracts from the EPG contract master.
Step 19	bridge-domain member <i>bd-name</i> Example: <pre>apic1(config-tenant-app-epg)# bridge-domain member T1BD1</pre>	Associates the EPG with the bridge domain.
Step 20	attribute-logical-expression <i>logical-expression</i> Example: <pre>apic1(config-tenant-app-epg)# attribute-logical-expression 'ip equals 192.168.103.30 force'</pre>	Adds a logical expression to the uSeg EPG as criteria.
Step 21	inherit-from-epg application <i>application-name</i> epg <i>EPG-contract-master-name</i> Example: <pre>apic1(config-tenant-app-epg)# inherit-from-epg application AP1 epg uSeg1_403_10</pre>	Configures this EPG to inherit contracts from the EPG contract master.
Step 22	exit Example: <pre>apic1(config-tenant-app-epg)# exit</pre>	Exits the configuration mode
Step 23	exit Example: <pre>apic1(config-tenant-app)# exit</pre>	Exits the configuration mode
Step 24	exit Example: <pre>apic1(config-tenant)# exit</pre>	Exits the configuration mode
Step 25	exit Example:	Exits the configuration mode

	Command or Action	Purpose
	apic1(config)# exit	

Example

```

ifav90-ifc1# show running-config tenant Tn1 application AP1
# Command: show running-config tenant Tn1 application AP1
# Time: Fri Apr 28 17:28:32 2017
tenant Tn1
  application AP1
    epg AEPg403
      bridge-domain member T1BD1
      contract consumer cctr5 imported
      contract provider T1ctrl_cif
      exit
    epg AEPg404
      bridge-domain member T1BD1
      inherit-from-epg application AP1 epg AEPg403
      exit
    epg uSeg1_403_10 type micro-segmented
      bridge-domain member T1BD1
      contract provider T1Ctrl_uSeg_l3out
      attribute-logical-expression 'ip equals 192.168.103.10 force'
      exit
    epg uSeg1_403_30 type micro-segmented
      bridge-domain member T1BD1
      attribute-logical-expression 'ip equals 192.168.103.30 force'
      inherit-from-epg application AP1 epg uSeg1_403_10
      exit
    exit
  exit
exit

```

Configuring L2Out EPG Contract Inheritance Using the NX-OS Style CLI

To configure contract inheritance for an external L2Out EPG, use the following commands:

Before you begin

Configure the tenant, VRF, and bridge-domain to be used by the EPGs.

Configure the Layer 2 outside network (L2Out) that the EPGs will use.

Configure the contracts to be shared by the EPGs, at the VRF level.

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **external-l2 epg** *external-l2-epg-name*
4. **bridge-domain member** *bd-name*
5. **contract provider** *contract-name* [*label label*]
6. **exit**
7. **external-l2 epg** *external-l2-epg-name*
8. **bridge-domain member** *bd-name*

9. **inherit-from-epg** *L2Out-contract-master-name*
10. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apic1(config)# tenant Tn1</code>	Creates or specifies the tenant to be configured; and enters into tenant configuration mode.
Step 3	external-l2 epg <i>external-l2-epg-name</i> Example: <code>apic1(config-tenant)# external-l2 epg l2out1:l2Ext1</code>	Configures or specifies an external L2Out EPG. In this example, this is the L2out contract master.
Step 4	bridge-domain member <i>bd-name</i> Example: <code>apic1(config-tenant-l2ext-epg)# bridge-domain member T1BD1</code>	Associates the L2Out EPG with a bridge domain.
Step 5	contract provider <i>contract-name</i> [<i>label label</i>] Example: <code>apic1(config-tenant-l2ext-epg)# contract provider Tlctr_tcp</code>	Adds a contract to be provided by this EPG.
Step 6	exit Example: <code>apic1(config-tenant-l2ext-epg)# exit</code>	Exits the configuration mode
Step 7	external-l2 epg <i>external-l2-epg-name</i> Example: <code>apic1(config-tenant)# external-l2 epg L2out12:l2Ext12</code>	Configures an external L2Out EPG. In this example, this is the EPG that inherits contracts from the L2out contract master.
Step 8	bridge-domain member <i>bd-name</i> Example: <code>apic1(config-tenant-l2ext-epg)# bridge-domain member T1BD1</code>	Associates the L2out EPG with the bridge domain.
Step 9	inherit-from-epg <i>L2Out-contract-master-name</i> Example: <code>apic1(config-tenant-l2ext-epg)# inherit-from-epg epg l2out1:l2Ext1</code>	Configures this EPG to inherit contracts from the L2Out contract master.

	Command or Action	Purpose
Step 10	exit Example: apic1(config-tenant-l2ext-epg)# exit	Exits the configuration mode

Example

The steps above are taken from the following example:

```
apic1# show running-config tenant Tn1 external-l2
# Command: show running-config tenant Tn1 external-l2
# Time: Thu May 11 13:10:14 2017
tenant Tn1
  external-l2 epg l2out1:l2Ext1
    bridge-domain member T1BD1
    contract provider T1ctr_tcp
    exit
  external-l2 epg l2out10:l2Ext10
    bridge-domain member T1BD10
    contract provider T1ctr_tcp
    exit
  external-l2 epg l2out11:l2Ext11
    bridge-domain member T1BD11
    contract provider T1ctr_udp
    exit
  external-l2 epg l2out12:l2Ext12
    bridge-domain member T1BD12
    inherit-from-epg epg l2out1:l2Ext1
    inherit-from-epg epg l2out10:l2Ext10
    inherit-from-epg epg l2out11:l2Ext11
    inherit-from-epg epg l2out2:l2Ext2
    inherit-from-epg epg l2out3:l2Ext3
    inherit-from-epg epg l2out4:l2Ext4
    inherit-from-epg epg l2out5:l2Ext5
    inherit-from-epg epg l2out6:l2Ext6
    inherit-from-epg epg l2out7:l2Ext7
    inherit-from-epg epg l2out8:l2Ext8
    inherit-from-epg epg l2out9:l2Ext9
    exit
  external-l2 epg l2out2:l2Ext2
    bridge-domain member T1BD2
    contract provider T1ctr_tcp
    exit
  external-l2 epg l2out3:l2Ext3
    bridge-domain member T1BD3
    contract provider T1ctr_tcp
    exit
  external-l2 epg l2out4:l2Ext4
    bridge-domain member T1BD4
    contract provider T1ctr_tcp
    exit
  external-l2 epg l2out5:l2Ext5
    bridge-domain member T1BD5
    contract provider T1ctr_tcp
    exit
  external-l2 epg l2out6:l2Ext6
    bridge-domain member T1BD6
    contract provider T1ctr_tcp
    exit
```

```

external-l2 epg l2out7:l2Ext7
  bridge-domain member T1BD7
  contract provider Tlctr_tcp
exit
external-l2 epg l2out8:l2Ext8
  bridge-domain member T1BD8
  contract provider Tlctr_tcp
exit
external-l2 epg l2out9:l2Ext9
  bridge-domain member T1BD9
  contract provider Tlctr_tcp
exit
exit

```

Configuring External L3Out EPG Contract Inheritance Using the NX-OS Style CLI

To configure contract inheritance for an external L3Out EPG, use the following commands:

Before you begin

Configure the tenant, VRF, and bridge-domain to be used by the EPGs.

Configure the Layer 3 outside network (L3Out) that the EPGs will use.

Configure the contracts to be shared by the EPGs, at the VRF level.

Procedure

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: apic1(config)# tenant Tn1	Creates or specifies the tenant to be configured; and enters into tenant configuration mode.
Step 3	external-l3 epg <i>external-l3-epg-name</i> l3out <i>l3out-name</i> Example: apic1(config-tenant-app)# external-l3 epg l3Ext108 l3out T1L3out1	Configures an external L3Out EPG. In this example, this is the L3out contract master.
Step 4	vrf member <i>vrf-name</i> Example: apic1(tenant-l3out)# vrf member T1ctx1	Associates the L3out with the VRF.
Step 5	match ip <i>ip-address-and-mask</i> Example: apic1(config-tenant-l3ext-epg)# match ip 192.168.110.0/24 shared	Adds a subnet that identifies hosts as part of the EPG and adds the optional shared scope for the subnet.

	Command or Action	Purpose
Step 6	contract provider <i>contract-name</i> [label <i>label</i>] Example: <pre>apic1(config-tenant-l3ext-epg)# contract provider T1ctrl-L3out</pre>	Adds a contract to be provided by this EPG.
Step 7	exit Example: <pre>apic1(config-tenant-l3ext-epg)# exit</pre>	Exits the configuration mode
Step 8	external-l3 epg <i>external-l3-epg-name</i> l3out <i>l3out-name</i> Example: <pre>apic1(config-tenant-app)# external-l3 epg l3Ext110 l3out T1L3out1</pre>	Configures an external L3Out EPG. In this example, this is the EPG that inherits contracts from the L3out contract master.
Step 9	vrf member <i>vrf-name</i> Example: <pre>apic1(tenant-l3out)# vrf member T1ctx1</pre>	Associates the L3out with the VRF.
Step 10	match ip <i>ip-address-and-mask</i> Example: <pre>apic1(config-tenant-l3ext-epg)# match ip 192.168.112.0/24 shared</pre>	Adds a subnet that identifies hosts as part of the EPG and adds the optional shared scope for the subnet.
Step 11	inherit-from-epg <i>L3Out-contract-master-name</i> Example: <pre>apic1(config-tenant-l3ext-epg)# inherit-from-epg l3Ext108</pre>	Configures this EPG to inherit contracts from the L3Out contract master.
Step 12	exit Example: <pre>apic1(config-tenant-l3ext-epg)# exit</pre>	Exits the configuration mode

Example

```
ifav90-ifc1# show running-config tenant Tn1 external-l3 epg l3Ext110
# Command: show running-config tenant Tn1 external-l3 epg l3Ext110
# Time: Fri Apr 28 17:36:15 2017
tenant Tn1
  external-l3 epg l3Ext108 l3out T1L3out1
    vrf member T1ctx1
    match ip 192.168.110.0/24 shared
    contract provider T1ctrl-L3out
  exit
  external-l3 epg l3Ext110 l3out T1L3out1
    vrf member T1ctx1
    match ip 192.168.112.0/24 shared
    inherit-from-epg epg l3Ext108
  exit
exit
```

Configuring Contract Preferred Groups

About Contract Preferred Groups

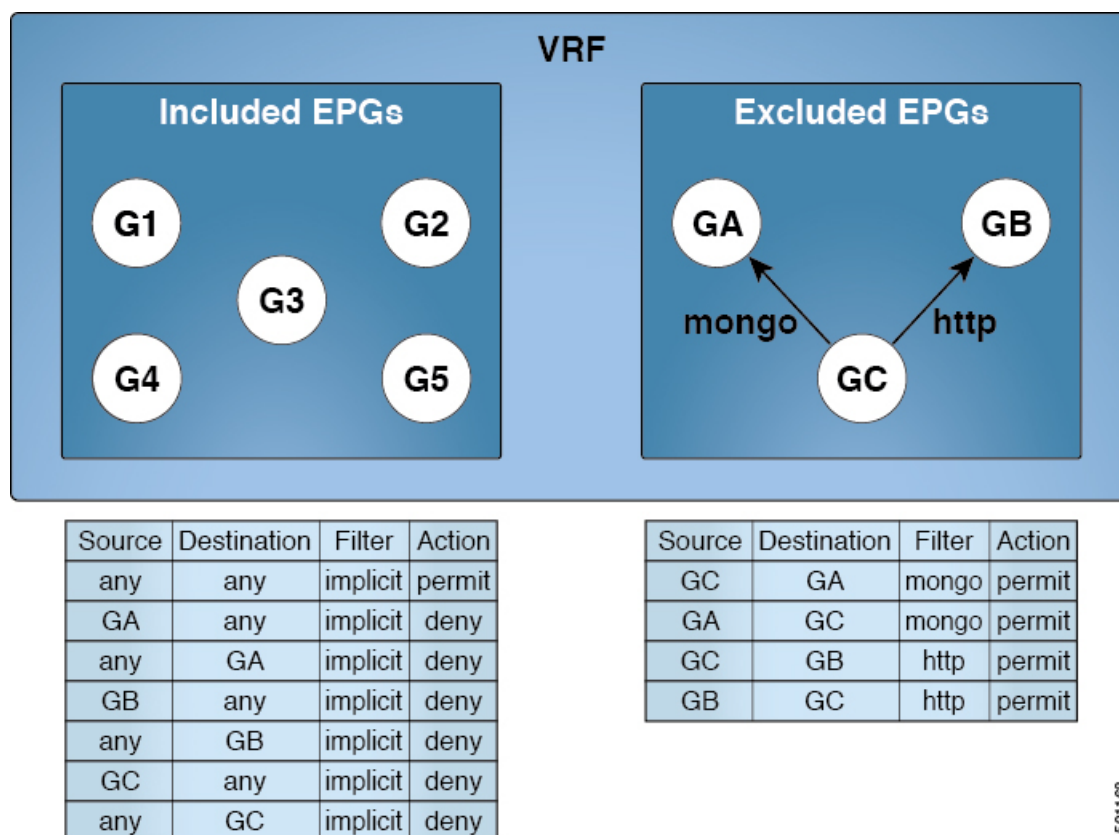
There are two types of policy enforcements available for EPGs in a VRF with a contract preferred group configured:

- **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 4: 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 workaround this issue, you can do one of the following:

- Disable and reenabte 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 .

Configuring Contract Preferred Groups Using the NX-OS Style CLI

You can use the APIC NX-OS style CLI to configure a contract preferred group. In this example, a contract preferred group is configured for a VRF. One of the EPGs using the VRF is included in the preferred group.

Before you begin

Create the tenants, VRFs, and EPGs that will consume the contract preferred group.

SUMMARY STEPS

1. `configure`
2. `tenant` *tenant-name*
3. `vrf context` *vrf-name*
4. `whitelist-blacklist-mix`
5. `bridge-domain` *bd-name*
6. `vrf member` *vrf-name*
7. `application` *app-name*
8. `epg` *epg-name*
9. `bridge-domain member` *bd-name*
10. `vrf-blacklist-mode`

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <pre>apic1# configure apic1(config)#</pre>	Enters configuration mode
Step 2	tenant <i>tenant-name</i> Example: <pre>apic1(config)# tenant tenant64</pre>	Creates a tenant or enters into tenant configuration mode
Step 3	vrf context <i>vrf-name</i> Example: <pre>apic1(config-tenant)# vrf context vrf64</pre>	Creates a VRF or enters into VRF configuration mode.
Step 4	whitelist-blacklist-mix Example: <pre>apic1(config-tenant-vrf)# whitelist-blacklist-mix apic1(config-tenant-vrf)# exit</pre>	Enables a contract preferred group for the VRF and then returns to tenant configuration mode.
Step 5	bridge-domain <i>bd-name</i> Example: <pre>apic1(config-tenant)# bridge-domain bd64</pre>	Creates a bridge-domain for the VRF or enters into BD configuration mode.
Step 6	vrf member <i>vrf-name</i> Example: <pre>apic1(config-tenant-bd)# vrf member vrf64 apic1(config-tenant-bd)# exit</pre>	Associates the VRF with the bridge-domain and returns to tenant configuration mode.
Step 7	application <i>app-name</i> Example: <pre>apic1(config-tenant)# application app-ldap</pre>	Creates an application or enters into application configuration mode.
Step 8	epg <i>epg-name</i> Example: <pre>apic1(config-tenant-app)# epg epg-ldap</pre>	Creates an EPG or enters into EPG tenant-app EPG configuration mode.
Step 9	bridge-domain member <i>bd-name</i> Example: <pre>apic1(config-tenant-app-epg)# bridge-domain member bd64</pre>	Associates the EPG with the bridge-domain .
Step 10	vrf-blacklist-mode Example: <pre>apic1(config-tenant-app-epg)# vrf-blacklist-mode</pre>	Configures this EPG to be included in the contract preferred group.

Example

The following example creates a contract preferred group for `vrf64` and includes `epg-ldap` in it.

```
apic1# configure
apic1(config)# tenant tenant64
apic1(config-tenant)# vrf context vrf64
apic1(config-tenant-vrf)# whitelist-blacklist-mix
apic1(config-tenant-vrf)# exit

apic1(config-tenant)# bridge-domain bd64
apic1(config-tenant-bd)# vrf member vrf64
apic1(config-tenant-bd)# exit

apic1(config-tenant)# application app-ldap
apic1(config-tenant-app)# epg epg-ldap
apic1(config-tenant-app-epg)# bridge-domain member bd64
apic1(config-tenant-app-epg)# vrf-blacklist-mode
```

Exporting a Contract to Another Tenant

You can export a contract from one tenant and import it to another. In the tenant that imports the contract, the contract can be applied only as a consumer contract. The contract can be renamed during the export.

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **contract** *contract-name*
4. **scope** {**application** | **exportable** | **tenant** | **vrf**}
5. **export to tenant** *other-tenant-name* **as** *new-contract-name*
6. **exit**
7. **exit**
8. **tenant** *tenant-name*
9. **application** *app-name*
10. **epg** *epg-name*
11. **contract consumer** *consumer-contract-name* **imported**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apic1(config)# tenant RedCorp</code>	Enters the tenant configuration mode for the exporting tenant.

	Command or Action	Purpose
Step 3	contract <i>contract-name</i> Example: <code>apic1(config-tenant)# contract web80</code>	Enters the contract configuration mode for the contract to be exported.
Step 4	scope { application exportable tenant vrf } Example: <code>apic1(config-tenant-contract)# scope exportable</code>	Configures how the contract can be shared. The scope can be: • application—Can be shared among the EPGs of the same application. • exportable—Can be shared across tenants. • tenant—Can be shared among the EPGs of the same tenant. • vrf—Can be shared among the EPGs of the same VRF.
Step 5	export to tenant <i>other-tenant-name</i> as <i>new-contract-name</i> Example: <code>apic1(config-tenant-contract)# export to tenant BlueCorp as webContract1</code>	Exports the contract to the other tenant. You can use the same contract name or you can rename it.
Step 6	exit Example: <code>apic1(config-tenant-contract)# exit</code>	Returns to the tenant configuration mode.
Step 7	exit Example: <code>apic1(config-tenant)# exit</code>	Returns to the global configuration mode.
Step 8	tenant <i>tenant-name</i> Example: <code>tenant BlueCorp</code>	Enters the tenant configuration mode for the importing tenant.
Step 9	application <i>app-name</i> Example: <code>apic1(config-tenant)# application BlueStore</code>	Enters application configuration mode.
Step 10	epg <i>epg-name</i> Example: <code>apic1(config-tenant-app)# epg BlueWeb</code>	Enters configuration mode for the EPG to be linked to the contract.
Step 11	contract consumer <i>consumer-contract-name</i> imported Example: <code>apic1(config-tenant-app-epg)# contract consumer webContract1 imported</code>	Specifies the imported consumer contract for this EPG. The endpoints in this EPG may initiate communication with any endpoint in an EPG that is providing this contract.

Examples

This example shows how to export a contract from the tenant RedCorp to the tenant BlueCorp, where it will be a consumer contract.

```
apic# configure
apic1(config)# tenant RedCorp
apic1(config-tenant)# contract web80
apic1(config-tenant-contract)# scope exportable
apic1(config-tenant-contract)# export to tenant BlueCorp as webContract1
apic1(config-tenant-contract)# exit
apic1(config-tenant)# exit
apic1(config)# tenant BlueCorp
apic1(config-tenant)# application BlueStore
apic1(config-tenant-application)# epg BlueWeb
apic1(config-tenant-application-epg)# contract consumer webContract1 imported
```

Configuring Contract or 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	<code><vzException consRegex="common" field="Tenant" name="excep03" provRegex="t1" /></code>	This example, excludes EPGs using the <code>common</code> tenant from consuming contracts provided by the <code>t1</code> tenant.
VRF	<code><vzException consRegex="ctx1" field="Ctx" name="excep05" provRegex="ctx1" /></code>	This example excludes members of <code>ctx1</code> from consuming the services provided by the same VRF.
EPG	<code><vzException consRegex="EPgPa*" field="EPg" name="excep03" provRegex="EPg03" /></code>	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	<code><vzException consRegex="uni/tn-t36/ap-customer/epg-epg193" field="Dn" name="excep04" provRegex="uni/tn-t36/ap-customer/epg-epg200" /></code>	This example excludes <code>epg193</code> from consuming the contract provided by <code>epg200</code> .
Tag	<code><vzException consRegex="red" field="Tag" name="excep01" provRegex="green" /></code>	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 NX-OS Style CLI

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 Configure filters for HTTP and HTTPS, using commands as in the following example:

Example:

```
apic1(config)# tenant t2
apic1(config-tenant)# access-list acl
apic1(config-tenant-acl)# match ip
apic1(config-tenant-acl)# match tcp dest 80
apic1(config-tenant-acl)# exit
apic1(config-tenant)# access-list ac2
apic1(config-tenant-acl)# match ip
apic1(config-tenant-acl)# match tcp dest 443
```

Step 2 Configure a contract that excludes EPg01 from consuming it and EPg03 from providing it.

Example:

```
apic1(config-tenant)# contract webCtrct
apic1(config-tenant-contract)# subject https-subject
apic1(config-tenant-contract-subj)# exception name EPG consumer-regexp EPg01 field EPg provider-regexp
EPg03
apic1(config-tenant-contract-subj)# access-group ac1 in blacklist
apic1(config-tenant-contract-subj)# access-group ac2 in whitelist
```

Creating Quota Management

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 NX-OS Style CLI

This procedure explains how to create a quota management configuration using the NX-OS Style CLI.

SUMMARY STEPS

1. Create a quota management configuration using the NX-OS CLI:

DETAILED STEPS

Create a quota management configuration using the NX-OS CLI:

Example:

```
apic1# conf t
apic1(config)# quota fvBD max 100 scope uni/tn-green exceed-action fault
apic1(config)# quota fvBD max 1000 scope uni exceed-action fail
apic1(config)# quota fvBD max 34 tenant red
```

Syntax:

```
[no] quota <className> max <maxValue> [exceed-action {fail|fault}] \
[scope <containerDn>| tenant <tenantName> \
[bridge-domain <bd>|application <ap> [epg <epgName>]]}]
```

where *<className>* is the managed object className such as **fvBD** or **fvCtx** etc. All the eligible classes accordingly to the presence of the quota flag in the model are accepted.

where *<maxValue>* is the value after which the *<exceed-action>* is applied.

where *<exceed-action>* is the action to be taken after the *<maxValue>* is exceeded, can either be:

- **fail**: when you want to fail the transaction exceeding the limit.
- **fault**: raise a fault.

where *<containerDn>* is the tree under which the limit will be enforced. "uni" will be across the whole ACI policy model, "tenant green" will be for the tenant green.



CHAPTER 6

Configuring Layer 2 External Connectivity

- [Configuring Layer 2 External Connectivity, on page 77](#)
- [Configuring VLAN Domains, on page 81](#)
- [Configuring Q-in-Q Encapsulation Mapping for EPGs, on page 89](#)
- [Support Fibre Channel over Ethernet Traffic on the ACI Fabric, on page 91](#)
- [Fibre Channel NPV, on page 104](#)
- [Configuring 802.1Q Tunnels, on page 110](#)
- [Configuring Dynamic Breakout Ports, on page 116](#)
- [Configuring Port Profiles, on page 120](#)
- [Microsegmentation on Virtual Switches, on page 126](#)
- [Configuring Microsegmentation on Bare-Metal , on page 128](#)
- [Configuring Layer 2 IGMP Snoop Multicast, on page 131](#)
- [Configuring Port Security, on page 138](#)
- [Configuring Proxy ARP, on page 145](#)
- [Configuring Traffic Storm Control, on page 154](#)
- [Configuring MACsec, on page 156](#)

Configuring Layer 2 External Connectivity

Layer 2 External Connectivity represents the switching network between the ACI leaf switches (aka border leaf) and an External Router. The VLAN representing the external L2 network is mapped to one of the bridge-domains within the fabric, which provides the Layer 2 extension for the bridge-domain and lets the EPGs using the bridge-domain talk to the outside network. The outside network is mapped to an EPG, which helps in realizing contracts between different internal applications and different L2 outside VLANs across nodes.

**Caution**

Do not mix the GUI and the CLI, when doing per-interface configuration on APIC. Configurations performed in the GUI, may only partially work in the NX-OS CLI.

For example, if you configure a switch port in the GUI at **Tenants > *tenant-name* > Application Profiles > *application-profile-name* > Application EPGs > *EPG-name* > Static Ports > Deploy Static EPG on PC, VPC, or Interface**

Then you use the show running-config command in the NX-OS style CLI, you receive output such as:

```
leaf 102
interface ethernet 1/15
switchport trunk allowed vlan 201 tenant t1 application apl epg ep1
exit
exit
```

If you use these commands to configure a static port in the NX-OS style CLI, the following error occurs:

```
apic1(config)# leaf 102
apic1(config-leaf)# interface ethernet 1/15
apic1(config-leaf-if)# switchport trunk allowed vlan 201 tenant t1 application apl epg ep1
```

```
No vlan-domain associated to node 102 interface ethernet1/15 encap vlan-201
```

This occurs because the CLI has validations that are not performed by the APIC GUI. For the commands from the show running-config command to function in the NX-OS CLI, a vlan-domain must have been previously configured. The order of configuration is not enforced in the GUI.

The configuration for Layer2 external connectivity is similar to a static application EPG, where you map a VLAN on a node port to an EPG and map the EPG to a bridge-domain to provide/consume contracts.

SUMMARY STEPS

1. Access configuration mode.
2. Enter tenant configuration mode.
3. **[no] external-l2 epg *epg-name***
4. Assign a bridge domain to the EPG.
5. Return to tenant configuration mode.
6. Return to global configuration mode.
7. Specify the leaf to be configured.
8. Specify a port for the external EPG.
9. By default, a port is in Layer 2 trunk mode. If the port is in Layer 3 mode, convert it to Layer 2 trunk mode using this command.
10. Associate the interface with a VLAN domain.
11. Assigns a VLAN on the leaf port and maps the VLAN to a layer 2 external EPG, with the **switchport trunk allowed vlan *vlan-id* tenant *tenant-name* external-l2 epg *epg-name*** command.
12. Assign a VLAN on the leaf port and map the VLAN to an external SVI with the **switchport {trunk allowed | trunk native | access} vlan *vlan-id* tenant *tenant-name* external-svi** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Access configuration mode.	

	Command or Action	Purpose
	Example: <code>apic1# configure</code>	
Step 2	Enter tenant configuration mode. Example: <code>apic1(config)# tenant exampleCorp</code>	
Step 3	[no] external-l2 epg epg-name Example: <code>apic1(config-tenant)# external-l2 epg extendBD1</code>	Create (or delete) an external layer 2 EPG.
Step 4	Assign a bridge domain to the EPG. Example: <code>apic1(config-tenant-extl2epg)# bridge-domain member bd1</code>	
Step 5	Return to tenant configuration mode. Example: <code>apic1(config-tenant-extl2epg)# exit</code>	
Step 6	Return to global configuration mode. Example: <code>apic1(config-tenant)# exit</code>	
Step 7	Specify the leaf to be configured. Example: <code>apic1(config)# leaf 101</code>	
Step 8	Specify a port for the external EPG. Example: <code>apic1(config-leaf)# interface eth 1/2</code>	
Step 9	By default, a port is in Layer 2 trunk mode. If the port is in Layer 3 mode, convert it to Layer 2 trunk mode using this command. Example: <code>apic1(config-leaf-if)# switchport</code>	
Step 10	Associate the interface with a VLAN domain. Example: <code>apic1(config-leaf-if)# vlan-domain member dom1</code>	
Step 11	Assigns a VLAN on the leaf port and maps the VLAN to a layer 2 external EPG, with the switchport trunk allowed vlan vlan-id tenant tenant-name external-l2 epg epg-name command.	Note The interface must be associated with a VLAN domain or this command is rejected.

	Command or Action	Purpose
	Example: <pre>apic1(config-leaf-if)# switchport trunk allowed vlan 10 tenant exampleCorp external-l2 epg extendBD1</pre>	
Step 12	Assign a VLAN on the leaf port and map the VLAN to an external SVI with the switchport {trunk allowed trunk native access} vlan <i>vlan-id</i> tenant <i>tenant-name</i> external-svi command. Example: <pre>apic1(config-leaf-if)# switchport trunk allowed vlan 10 tenant exampleCorp external-svi</pre>	Note The interface must be associated with a VLAN domain or this command is rejected.

Examples

This example shows how to deploy a layer 2 port for external connectivity.

```
apic1# configure
apic1(config)# tenant exampleCorp
apic1(config-tenant)# external-l2 epg extendBD1
apic1(config-tenant-extl2epg)# bridge-domain member bd1
apic1(config-tenant-extl2epg)# exit
apic1(config-tenant)# exit

apic1(config)# leaf 101
apic1(config-leaf)# interface eth 1/2
apic1(config-leaf-if)# switchport
apic1(config-leaf-if)# switchport mode trunk
apic1(config-leaf-if)# switchport trunk allowed vlan 10 tenant exampleCorp external-l2 epg
extendBD1
```

This example shows how to deploy a layer 2 port channel or vPC for external connectivity.

...

```
apic1(config)# leaf 101
apic1(config-leaf)# interface port-channel po1
apic1(config-leaf-if)# switchport trunk allowed vlan 10 tenant exampleCorp external-l2 epg
extendBD1
```

These examples show how to configure SVI on a layer 2 interface for external connectivity.

```
apic1(config)# leaf 101

apic1(config-leaf)# interface ethernet 1/5
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk allowed vlan 10 tenant exampleCorp external-svi
apic1(config-leaf-if)# no switchport trunk allowed vlan 10 tenant exampleCorp external-svi

apic1(config-leaf)# interface ethernet 1/37
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport access vlan 11 tenant exampleCorp external-svi
```

```
apic1(config-leaf-if)# no switchport access vlan 11 tenant exampleCorp external-svi

apic1(config-leaf)# interface port-channel po34
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk native vlan 12 tenant exampleCorp external-svi
apic1(config-leaf-if)# no switchport trunk native vlan 12 tenant exampleCorp external-svi
```

Configuring VLAN Domains

About VLAN Domains

ACI fabric can be partitioned into groups of 4K VLANs to allow a large number of layer 2 domains across the fabric, which can be used by multiple tenants. A VLAN domain represents a set of VLANs that can be configured on group of nodes and ports. VLAN domains let multiple tenants share and independently manage common fabric resources such as nodes, ports, and VLANs. A tenant can be provided access to one or more VLAN domains. For more information about VLAN pools, see *Endpoint Groups* in the *ACI Policy Model* chapter in *Cisco Application Centric Infrastructure Fundamentals*.

VLAN domains can be static or dynamic. Static VLAN domains support static VLAN pools, while dynamic VLAN domains can support both static and dynamic VLAN pools. VLANs in static pools are managed by the user and are used for applications such as connectivity to bare metal hosts. VLANs in the dynamic pool are allocated and managed by the APIC without user intervention and are used for applications such as VMM. The default type for VLAN domains and VLAN pools within the domain is static.

The fabric administrator performs the following steps before tenants can start using the fabric resources for their L2/L3 configurations:

1. Create VLAN domains and assign VLANs in each VLAN domain.
2. Assign the external facing ports on the leaf switches to one or more VLAN domains.
3. Convert a port to L2/L3 by using the **[no] switchport** command. The default state of a port is L2(switchport) in trunk mode.
4. For an L2 port, set the scope of a VLAN on a port to be global or local. The default is global.

The fabric administrator can update any configuration in these steps even after VLAN domains are assigned to tenants and are in use by tenant applications.

A Note About Spanning Tree and VLAN Domains

Although the ACI fabric does not participate in spanning tree, it can partition a spanning tree domain based on access policy configuration. ACI does not rely on a bridge domain or its settings to determine spanning tree domains. Instead, leaf switches flood BPDUs within the same VLAN encapsulation, if a VLAN Pool is assigned to EPG domains. The VLAN pool assigned to EPG domains ultimately serves as the spanning tree domain.

Using multiple EPG domains tied to different VLAN Pools does not allow BPDUs to flood across endpoints properly, even if they are all using the same VLAN ID. The type of EPG domain, (physical or Layer 2 external), does not change this behavior.

Because the ACI Fabric floods all BPDUs from all devices within a spanning-tree domain, this may trigger behaviors on external devices that are verifying BPDU info, such as the MAC address per interface. An example of a feature that activates is "spanning-tree EtherChannel misconfig guard" found on IOS devices. These features should be taken into account when utilizing ACI as a Layer 2 Tunnel.

**Note**

Multiple Spanning Tree (MST) is not supported on interfaces configured with the Per Port VLAN feature (configuring multiple EPGs on a leaf switch using the same VLAN ID with localPort scope).

Basic VLAN Domain Configuration

SUMMARY STEPS

1. **configure**
2. **[no] vlan-domain *domain-name* [dynamic]**
3. **[no] vlan *range* [dynamic]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] vlan-domain <i>domain-name</i> [dynamic] Example: <code>apic1(config)# vlan-domain dom2 dynamic</code>	Creates a VLAN domain or edits an existing domain. Include the dynamic keyword to create a dynamic VLAN pool. The default is static.
Step 3	[no] vlan <i>range</i> [dynamic] Example: <code>apic1(config-vlan)# vlan 1000-1999,4001</code>	Assigns a range or a comma-separated list of VLANs to the VLAN domain. A VLAN can be either static or dynamic. A static VLAN is configured by the user, such as for providing connectivity from a host to an external switched network, while VLANs in the dynamic range are configured internally by an APIC application, such as a VMM or L4-L7 services. The default type is static. Note A static domain cannot contain dynamic VLANs. A VLAN on a given port can map to only one vlan-domain. This is enforced during configuration.

Examples

This example shows how to configure basic VLAN domains.

```
apic1# configure
apic1(config)# vlan-domain dom1
apic1(config-vlan)# vlan 1000-1999,4001
apic1(config-vlan)# exit
apic1(config)# vlan-domain dom2 dynamic
apic1(config-vlan)# vlan 101-200
apic1(config-vlan)# vlan 301-400 dynamic
```

Advanced VLAN Domain Configuration

SUMMARY STEPS

1. **configure**
2. **[no] vlan-domain** *domain-name* [**dynamic**] [**type** {**phys** | **l2ext** | **l3ext**}]
3. **[no] vlan-pool** *vlan-pool-name*
4. **[no] vlan range** [**dynamic**]
5. **show vlan-domain** [**name** *domain-name*] [**vlan** *vlan-id*] [**leaf** *leaf-id*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	[no] vlan-domain <i>domain-name</i> [dynamic] [type { phys l2ext l3ext }] Example: apic1(config)# vlan-domain dom1 type phys	Creates a VLAN domain or edits an existing domain. Include the dynamic keyword to create a dynamic VLAN pool. The default is static. The type option is visible and mandatory if one or more of the following conditions exist: • If all three vlan-domain types are not present for this domain name • If the three vlan-domain types have different VLAN pools • If the three vlan-domain types share the same VLAN pool but if the pool name differs from the vlan-domain name
Step 3	[no] vlan-pool <i>vlan-pool-name</i> Example:	Creates a VLAN pool. This command is available only when the type option is present in the vlan-domain

	Command or Action	Purpose
	<code>apic1(config-leaf) # vlan-pool myVlanPool3</code>	command. You must declare the VLAN pool before adding VLANs with the vlan command.
Step 4	[no] vlan range [dynamic] Example: <pre>apic1(config-vlan-domain) # vlan 1000-1999,4001</pre>	Assigns a range or a comma-separated list of VLANs to the VLAN domain. A VLAN can be either static or dynamic. A static VLAN is configured by the user, such as for providing connectivity from a host to an external switched network, while VLANs in the dynamic range are configured internally by an APIC application, such as a VMM or L4-L7 services. The default type is static. Note A static domain cannot contain dynamic VLANs. A VLAN on a given port can map to only one vlan-domain. This is enforced during configuration.
Step 5	show vlan-domain [name domain-name] [vlan vlan-id] [leaf leaf-id] Example: <pre>apic1(config-vlan-domain) # show vlan-domain name dom1 vlan 1002 leaf 102</pre>	Displays vlan-domain usage for applications such as App-EPG, sub-interface, external SVI, and external-L2.

Examples

This example shows how to configure a VLAN domain with a VLAN pool.

```
apic1# configure
(config) # vlan-domain dom1 type phys
(config-vlan-domain) # vlan-pool myVlanPool3
(config-vlan-domain) # vlan 1000-1999, 4001
```

Associating a VLAN Domain to a Port

SUMMARY STEPS

1. **configure**
2. **leaf node-id1-node-id2**
3. **interface type**
4. **[no] vlan-domain member domain-name**
5. **[no] switchport**
6. (Optional) **[no] switchport vlan scope local**
7. **show vlan-domain [name domain-name] [vlan vlan-id] [leaf leaf-id]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	leaf node-id1-node-id2 Example: apic1(config)# leaf 101-102	Specifies the pair of leafs to be configured.
Step 3	interface type Example: apic1(config-leaf)# int eth 1/1-24	Specifies a port or range of ports to be associated with the VLAN domain.
Step 4	[no] vlan-domain member domain-name Example: apic1(config-leaf-if)# vlan-domain member dom1	Assigns the specified ports to the VLAN domain.
Step 5	[no] switchport Example: apic1(config-leaf-if)# switchport	By default, a port is in Layer 2 trunk mode. If the port is in Layer 3 mode, it must be converted to Layer 2 trunk mode using this command.
Step 6	(Optional) [no] switchport vlan scope local Example: apic1(config-leaf-if)# switchport vlan scope local	By default, the scope of a VLAN is global to the node. One VLAN can be mapped to only one EPG in the node. When the VLAN scope is local to the port, the mapping from VLAN to EPG can be different for different ports on the same node. To return the scope to global, use the no command prefix.
Step 7	show vlan-domain [name domain-name] [vlan vlan-id] [leaf leaf-id] Example: apic1(config-leaf-if)# show vlan-domain name dom1 vlan 1002 leaf 102	Displays vlan-domain usage for applications such as App-EPG, external SVI, and external-L2.

Examples

This example shows how to associate a VLAN domain to ports.

```
apic1# configure
(config) # leaf 101-102
(config-leaf) # int eth 1/1-24
(config-leaf-if) # vlan-domain member dom1

(config-leaf) # int eth 1/1-12
(config-leaf-if) # no switchport
(config-leaf) # int eth 1/13-24
```

```

(config-leaf-if) # switchport

(config) # leaf 101-102
(config-leaf) # int eth 1/1-12
(config-leaf-if) # switchport vlan scope local

(config-leaf) # int eth 1/13
(config-leaf-if) # no switchport vlan scope local

```

Associating a VLAN Domain to a Port-Channel

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id1-node-id2*
3. **interface port-channel** *port-channel-name*
4. **[no] vlan-domain member** *domain-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	leaf <i>node-id1-node-id2</i> Example: apic1(config)# leaf 101-102	Specifies the pair of leafs to be configured.
Step 3	interface port-channel <i>port-channel-name</i> Example: apic1(config-leaf)# int port-channel pc1	Specifies a port-channel to be associated with the VLAN domain.
Step 4	[no] vlan-domain member <i>domain-name</i> Example: apic1(config-leaf-if)# vlan-domain member dom1	Assigns the specified port-channel to the VLAN domain.

Examples

```

apic1# configure
apic1(config)# leaf 101-102
apic1(config-leaf)# int port-channel pc1
apic1(config-leaf-if)# vlan-domain member dom1

```

Associating a VLAN Domain to a Template Policy-Group

SUMMARY STEPS

1. **configure**
2. **template policy-group** *policy-group-name*
3. **[no] vlan-domain member** *domain-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	template policy-group <i>policy-group-name</i> Example: <code>apic1(config)# template policy-group myPolGp5</code>	Specifies the template policy-group to be configured.
Step 3	[no] vlan-domain member <i>domain-name</i> Example: <code>apic1(config-pol-grp-if)# vlan-domain member dom1</code>	Assigns the specified template policy-group to the VLAN domain.

Examples

```
apic1# configure
apic1(config)# template policy-group myPolGp5
apic1(config-pol-grp-if)# vlan-domain member dom1
```

Associating a VLAN Domain to a Template Port-Channel

SUMMARY STEPS

1. **configure**
2. **template port-channel** *policy-group-name*
3. **[no] vlan-domain member** *domain-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.

	Command or Action	Purpose
Step 2	template port-channel <i>policy-group-name</i> Example: <code>apic1(config)# template port-channel myPC7</code>	Specifies the template port-channel to be configured.
Step 3	[no] vlan-domain member <i>domain-name</i> Example: <code>apic1(config-if)# vlan-domain member dom1</code>	Assigns the specified template port-channel to the VLAN domain.

Examples

```
apic1# configure
apic1(config)# template port-channel myPC7
apic1(config-po-ch-if)# vlan-domain member dom1
```

Associating a VLAN Domain to a Virtual Port-Channel

SUMMARY STEPS

1. **configure**
2. **vpc context leaf** *node-id1 node-id2* [**fex** *fex-id1 fex-id2*]
3. **interface vpc** *vpc-name* [**fex** *fex-id1 fex-id2*]
4. **[no] vlan-domain member** *domain-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	vpc context leaf <i>node-id1 node-id2</i> [fex <i>fex-id1 fex-id2</i>] Example: <code>apic1(config)# vpc context leaf 101 102</code>	Specifies the VPC and leafs to be configured.
Step 3	interface vpc <i>vpc-name</i> [fex <i>fex-id1 fex-id2</i>] Example: <code>apic1(config-vpc)# int vpc vpc1</code>	Specifies a port-channel to be associated with the VLAN domain.
Step 4	[no] vlan-domain member <i>domain-name</i> Example: <code>apic1(config-vpc-if)# vlan-domain member dom1</code>	Assigns the specified VPC to the VLAN domain.

Examples

```
apic1# configure
apic1(config)# vpc context leaf 101 102
apic1(config-vpc)# int vpc vpc1
apic1(config-vpc-if)# vlan-domain member dom1
```

Configuring Q-in-Q Encapsulation Mapping for EPGs

Q-in-Q Encapsulation Mapping for EPGs

Using Cisco 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 ACI switch. Ingressing single-tagged and untagged traffic is dropped.

This feature is only supported on 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, 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.

The following features and options are not supported with this feature:

- Per-Port VLAN feature
- FEX connections
- Mixed Mode is not supported. 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 is not supported

- Per-vlan MCP is not supported between ports in Q-in-Q encapsulation mode and ports in regular trunk mode.
- When VPC ports are enabled for Q-in-Q encapsulation mode, VLAN consistency checks are not performed.

Mapping EPGs to Q-in-Q Encapsulated Leaf Interfaces Using the NX-OS Style CLI

Enable an interface for Q-in-Q encapsulation and associate the interface with an EPG.

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. **Configure**
2. **leaf** *number*
3. **interface ethernet***slot/port*
4. **switchport mode dot1q-tunnel** *doubleQtagPort*
5. **switchport trunk qinq outer-vlan***vlan-number* **inner-vlan** *vlan-number* **tenant** *tenant-name* **application** *application-name* **epg** *epg-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	leaf <i>number</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	interface ethernet <i>slot/port</i> Example: <code>apic1 (config-leaf)# interface ethernet 1/25</code>	Specifies the interface to be configured.
Step 4	switchport mode dot1q-tunnel <i>doubleQtagPort</i> Example: <code>apic1(config-leaf-if)# switchport mode dot1q-tunnel doubleQtagPort</code>	Enables an interface for Q-in-Q encapsulation.
Step 5	switchport trunk qinq outer-vlan <i>vlan-number</i> inner-vlan <i>vlan-number</i> tenant <i>tenant-name</i> application <i>application-name</i> epg <i>epg-name</i>	Associates the interface with an EPG.

	Command or Action	Purpose
	Example: <pre>apic1(config-leaf-if)# switchport trunk qinq outer-vlan 202 inner-vlan 203 tenant tenant64 application AP64 epg EPG64</pre>	

Example

The following example enables Q-in-Q encapsulation (with outer-VLAN ID 202 and inner-VLAN ID 203) on the leaf interface 101/1/25, and associates the interface with EPG64.

```
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/25
apic1(config-leaf-if)#switchport mode dot1q-tunnel doubleQtagPort
apic1(config-leaf-if)# switchport trunk qinq outer-vlan 202 inner-vlan 203 tenant tenant64
application AP64 epg EPG64
```

Support Fibre Channel over Ethernet Traffic on the ACI Fabric

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.



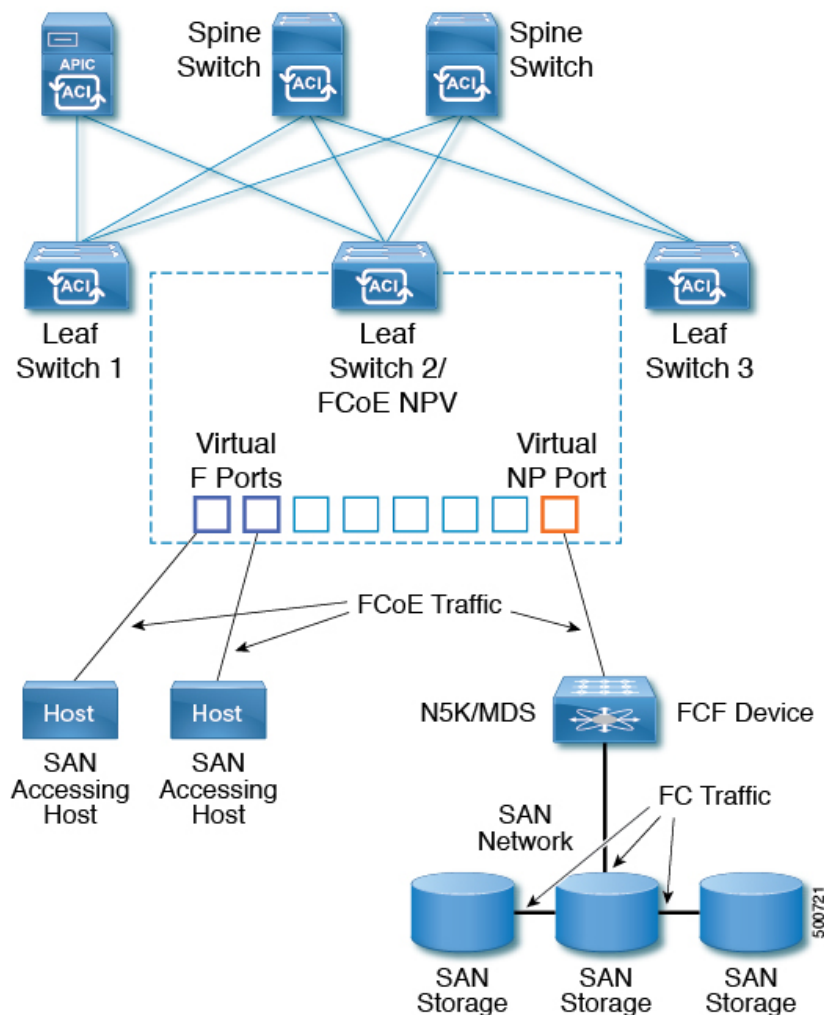
Note

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 5: 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.

FCoE Guidelines and Limitations

FCoE is supported on the following switches:

- N9K-C93180LC-EX (When 40 Gigabit Ethernet (GE) ports are enabled as FCoE F or NP ports, they cannot be enabled for 40GE port breakout. FCoE is not supported on breakout ports.)
- N9K-C93108TC-EX
- N9K-C93180YC-EX
- N9K-C93108TC-FX (FCoE support on FEX ports)
- N9K-C93180YC-FX (FCoE support on FEX ports, 40G ports (1/49-54), and 4x10G breakout ports)

FCoE is supported on the following Nexus FEX devices:

- N2K-C2348UPQ-10GE
- N2K-C2348TQ-10GE
- N2K-C2232PP-10GE
- N2K-B22DELL-P
- N2K-B22HP-P
- N2K-B22IBM-P
- N2K-B22DELL-P-FI

The vlan used for FCoE should have vlanScope set to Global. vlanScope set to portLocal is not supported for FCoE. The value is set via the L2 Interface Policy l2IfPol.

FCoE NX-OS Style CLI Configuration

Configuring FCoE Connectivity Without Policies or Profiles Using the NX-OS Style CLI

The following sample NX-OS style CLI sequences configure FCoE connectivity for EPG **e1** under tenant **t1** without configuring or applying switch-level and interface-level policies and profiles.

Procedure

	Command or Action	Purpose
Step 1	Under the target tenant configure a bridge domain to support FCoE traffic. Example: <pre>apicl(config)# tenant t1 apicl(config-tenant)# vrf context v1 apicl(config-tenant-vrf)# exit apicl(config-tenant)# bridge-domain b1 apicl(config-tenant-bd)# fc apicl(config-tenant-bd)# vrf member v1 apicl(config-tenant-bd)# exit apicl(config-tenant)# exit</pre>	The sample command sequence creates bridge domain b1 under tenant t1 configured to support FCoE connectivity.
Step 2	Under the same tenant, associate the target EPG with the FCoE-configured bridge domain. Example: <pre>apicl(config)# tenant t1 apicl(config-tenant)# application a1 apicl(config-tenant-app)# epg e1 apicl(config-tenant-app-epg)# bridge-domain member b1 apicl(config-tenant-app-epg)# exit apicl(config-tenant-app)# exit apicl(config-tenant)# exit</pre>	The sample command sequence creates EPG e1 and associates that EPG with the FCoE-configured bridge domain b1 .
Step 3	Create a VSAN domain, VSAN pools, VLAN pools and VSAN to VLAN mapping. Example: A <pre>apicl(config)# vsan-domain dom1 apicl(config-vsan)# vsan 1-10 apicl(config-vsan)# vlan 1-10 apicl(config-vsan)# fcoe vsan 1 vlan 1 loadbalancing src-dst-ox-id apicl(config-vsan)# fcoe vsan 2 vlan 2</pre> Example:	In Example A, the sample command sequence creates VSAN domain, dom1 with VSAN pools and VLAN pools, maps VSAN 1 to VLAN 1 and maps VSAN 2 to VLAN 2 In Example B, an alternate sample command sequence creates a reusable VSAN attribute template pol1 and then creates VSAN domain dom1, which inherits the attributes and mappings from that template.

	Command or Action	Purpose
	B <pre> apic1(config)# template vsan-attribute poll apic1(config-vsan-attr)# fcoe vsan 2 vlan 12 loadbalancing src-dst-ox-id apic1(config-vsan-attr)# fcoe vsan 3 vlan 13 loadbalancing src-dst-ox-id apic1(config-vsan-attr)# exit apic1(config)# vsan-domain dom1 apic1(config-vsan)# vsan 1-10 apic1(config-vsan)# vlan 1-10 apic1(config-vsan)# inherit vsan-attribute poll apic1(config-vsan)# exit </pre>	
Step 4	Create the physical domain to support the FCoE Initialization (FIP) process. Example: <pre> apic1(config)# vlan-domain fipVlanDom apic1(config-vlan)# vlan 120 apic1(config-vlan)# exit </pre>	In the example, the command sequence creates a regular VLAN domain, fipVlanDom , which includes VLAN 120 to support the FIP process.
Step 5	Under the target tenant configure a regular bridge domain. Example: <pre> apic1(config)# tenant t1 apic1(config-tenant)# vrf context v2 apic1(config-tenant-vrf)# exit apic1(config-tenant)# bridge-domain fip-bd apic1(config-tenant-bd)# vrf member v2 apic1(config-tenant-bd)# exit apic1(config-tenant)# exit </pre>	In the example, the command sequence creates bridge domain fip-bd .
Step 6	Under the same tenant, associate this EPG with the configured regular bridge domain. Example: <pre> apic1(config)# tenant t1 apic1(config-tenant)# application a1 apic1(config-tenant-app)# epg epg-fip apic1(config-tenant-app-epg)# bridge-domain member fip-bd apic1(config-tenant-app-epg)# exit apic1(config-tenant-app)# exit apic1(config-tenant)# exit </pre>	In the example, the command sequence associates EPG epg-fip with bridge domain fip-bd .
Step 7	Configure a VFC interface with F mode. Example: A <pre> apic1(config)# leaf 101 apic1(config-leaf)# interface ethernet 1/2 apic1(config-leaf-if)# vlan-domain member fipVlanDom apic1(config-leaf-if)# switchport trunk native vlan 120 tenant t1 application a1 epg epg-fip apic1(config-leaf-if)# exit </pre>	In example A the command sequence enables interface 1/2 on leaf switch 101 to function as an F port and associates that interface with VSAN domain dom1. Each of the targeted interfaces must be assigned one (and only one) VSAN in native mode. Each interface may be assigned one or more additional VSANs in regular mode. The sample command sequence associates the target interface 1/2 with:

	Command or Action	Purpose
	<pre> apicl(config-leaf)# exit apicl(config-leaf)# interface vfc 1/2 apicl(config-leaf-if)# switchport mode f apicl(config-leaf-if)# vsan-domain member dom1 apicl(config-leaf-if)# switchport vsan 2 tenant t1 application a1 epg e1 apicl(config-leaf-if)# switchport trunk allowed vsan 3 tenant t1 application a1 epg e2 apicl(config-leaf-if)# exit </pre> Example: B <pre> apicl(config)# vpc context leaf 101 102 apicl(config-vpc)# interface vpc vpc1 apicl(config-vpc-if)# vlan-domain member vfdom100 apicl(config-vpc-if)# vsan-domain member dom1 apicl(config-vpc-if)# #For FIP discovery apicl(config-vpc-if)# switchport trunk native vlan 120 tenant t1 application a1 epg epg-fip apicl(config-vpc-if)# switchport vsan 2 tenant t1 application a1 epg e1 apicl(config-vpc-if)# exit apicl(config-vpc)# exit apicl(config)# leaf 101-102 apicl(config-leaf)# interface ethernet 1/3 apicl(config-leaf-if)# channel-group vpc1 vpc apicl(config-leaf-if)# exit apicl(config-leaf)# exit </pre> Example: C <pre> apicl(config)# leaf 101 apicl(config-leaf)# interface vfc-po pc1 apicl(config-leaf-if)# vsan-domain member dom1 apicl(config-leaf-if)# switchport vsan 2 tenant t1 application a1 epg e1 apicl(config-leaf-if)# exit apicl(config-leaf)# interface ethernet 1/2 apicl(config-leaf-if)# channel-group pc1 apicl(config-leaf-if)# exit apicl(config-leaf)# exit </pre>	• VLAN 120 for FIP discovery and associates it with EPG epg-fip and application a1 under tenant t1. • VSAN 2 as a native VSAN and associates it with EPG e1 and application a1 under tenant t1. • VSAN 3 as a regular VSAN. In example B, the command sequence configures a vFC over a vPC with the same VSAN on both the legs. From the CLI you cannot specify different VSANs on each log. The alternate configuration can be carried out in the APIC advanced GUI.
Step 8	Configure a VFC interface with NP mode. Example: <pre> apicl(config)# leaf 101 apicl(config-leaf)# interface vfc 1/4 apicl(config-leaf-if)# switchport mode np apicl(config-leaf-if)# vsan-domain member dom1 </pre>	The sample command sequence enables interface 1/4 on leaf switch 101 to function as an NP port and associates that interface with VSAN domain dom1.
Step 9	Assign the targeted FCoE-enabled interfaces a VSAN. Example: <pre> apicl(config-leaf-if)# switchport trunk allowed vsan 1 tenant t1 application a1 epg e1 apicl(config-leaf-if)# switchport vsan 2 tenant t4 application a4 epg e4 </pre>	Each of the targeted interfaces must be assigned one (and only one) VSAN in native mode. Each interface may be assigned one or more additional VSANs in regular mode. The sample command sequence assigns the target interface to VSAN 1 and associates it with EPG e1 and application a1 under tenant t1. "trunk allowed" assigns vsan 1 regular

	Command or Action	Purpose
		mode status. The command sequence also assigns the interface a required native mode VSAN 2. As this example shows, it is permissible for different VSANs to provide different EPGs running under different tenants access to the same interfaces.

Configuring FCoE Connectivity With Policies and Profiles Using the NX-OS Style CLI

The following sample NX-OS style CLI sequences create and use policies to configure FCoE connectivity for EPG **e1** under tenant **t1**.

Procedure

	Command or Action	Purpose
Step 1	Under the target tenant configure a bridge domain to support FCoE traffic. Example: <pre> apic1# configure apic1(config)# tenant t1 apic1(config-tenant)# vrf context v1 apic1(config-tenant-vrf)# exit apic1(config-tenant)# bridge-domain b1 apic1(config-tenant-bd)# fc apic1(config-tenant-bd)# vrf member v1 apic1(config-tenant-bd)# exit apic1(config-tenant)# exit apic1(config)# </pre>	The sample command sequence creates bridge domain b1 under tenant t1 configured to support FCoE connectivity.
Step 2	Under the same tenant, associate your target EPG with the FCoE configured bridge domain. Example: <pre> apic1(config)# tenant t1 apic1(config-tenant)# application a1 apic1(config-tenant-app)# epg e1 apic1(config-tenant-app-epg)# bridge-domain member b1 apic1(config-tenant-app-epg)# exit apic1(config-tenant-app)# exit apic1(config-tenant)# exit apic1(config)# </pre>	The sample command sequence creates EPG e1 associates that EPG with FCoE-configured bridge domain b1 .
Step 3	Create a VSAN domain, VSAN pools, VLAN pools and VSAN to VLAN mapping. Example: A <pre> apic1(config)# vsan-domain dom1 apic1(config-vsan)# vsan 1-10 apic1(config-vsan)# vlan 1-10 </pre>	In Example A , the sample command sequence creates VSAN domain, dom1 with VSAN pools and VLAN pools, maps VSAN 1 VLAN 1 and maps VSAN 2 to VLAN 2 In Example B , an alternate sample command sequence creates a reusable vsan attribute template pol1 and then creates VSAN domain dom1 , which inherits the attributes and mappings from that template.

	Command or Action	Purpose
	<pre> apic1(config-vsan)# fcoe vsan 1 vlan 1 loadbalancing src-dst-ox-id apic1(config-vsan)# fcoe vsan 2 vlan 2 </pre> Example: B <pre> apic1(config)# template vsan-attribute poll apic1(config-vsan-attr)# fcoe vsan 2 vlan 12 loadbalancing src-dst-ox-id apic1(config-vsan-attr)# fcoe vsan 3 vlan 13 loadbalancing src-dst-ox-id apic1(config-vsan-attr)# exit apic1(config)# vsan-domain dom1 apic1(config-vsan)# inherit vsan-attribute poll apic1(config-vsan)# exit </pre>	
Step 4	Create the physical domain to support the FCoE Initialization (FIP) process. Example: <pre> apic1(config)# vlan-domain fipVlanDom apic1(config)# vlan-pool fipVlanPool </pre>	
Step 5	Configure a Fibre Channel SAN policy. Example: <pre> apic1# apic1# configure apic1(config)# template fc-fabric-policy ffp1 apic1(config-fc-fabric-policy)# fctimer e-d-tov 1111 apic1(config-fc-fabric-policy)# fctimer r-a-tov 2222 apic1(config-fc-fabric-policy)# fcoe fcmmap 0E:FC:01 apic1(config-fc-fabric-policy)# exit </pre>	The sample command sequence creates Fibre Channel SAN policy ffp1 to specify a combination of error-detect timeout values (EDTOV), resource allocation timeout values (RATOV), and the default FC map values for FCoE-enabled interfaces on a target leaf switch.
Step 6	Create a Fibre Channel node policy. Example: <pre> apic1(config)# template fc-leaf-policy flp1 apic1(config-fc-leaf-policy)# fcoe fka-adv-period 44 apic1(config-fc-leaf-policy)# exit </pre>	The sample command sequence creates Fibre Channel node policy flp1 to specify a combination of disruptive load-balancing enablement and FIP keep-alive values. These values also apply to all the FCoE-enabled interfaces on a target leaf switch.
Step 7	Create Node Policy Group. Example: <pre> apic1(config)# template leaf-policy-group lpg1 apic1(config-leaf-policy-group)# inherit fc-fabric-policy ffp1 apic1(config-leaf-policy-group)# inherit fc-leaf-policy flp1 apic1(config-leaf-policy-group)# exit </pre>	The sample command sequence creates a Node Policy group, lpg1 , which combines the values of the Fibre Channel SAN policy ffp1 and Fibre Channel node policy, flp1 . The combined values of this node policy group can be applied to Node profiles configured later.

	Command or Action	Purpose
	<pre>apicl(config)# exit apicl#</pre>	
Step 8	Create a Node Profile. Example: <pre>apicl(config)# leaf-profile lp1 apicl(config-leaf-profile)# leaf-group lg1 apicl(config-leaf-group)# leaf 101 apicl(config-leaf-group)# leaf-policy-group lpg1</pre>	The sample command sequence creates node profile lp1 associates it with node policy group lpg1 , node group lg1 , and leaf switch 101 .
Step 9	Create an interface policy group for F port interfaces. Example: <pre>apicl(config)# template policy-group ipg1 apicl(config-pol-grp-if)# priority-flow-control mode auto apicl(config-pol-grp-if)# switchport mode f apicl(config-pol-grp-if)# slow-drain pause timeout 111 apicl(config-pol-grp-if)# slow-drain congestion-timeout count 55 apicl(config-pol-grp-if)# slow-drain congestion-timeout action log</pre>	The sample command sequence creates interface policy group ipg1 and assigns a combination of values that determine priority flow control enablement, F port enablement, and slow-drain policy values for any interface that this policy group is applied to.
Step 10	Create an interface policy group for NP port interfaces. Example: <pre>apicl(config)# template policy-group ipg2 apicl(config-pol-grp-if)# priority-flow-control mode auto apicl(config-pol-grp-if)# switchport mode np apicl(config-pol-grp-if)# slow-drain pause timeout 111 apicl(config-pol-grp-if)# slow-drain congestion-timeout count 55 apicl(config-pol-grp-if)# slow-drain congestion-timeout action log</pre>	The sample command sequence creates interface policy group ipg2 and assigns a combination of values that determine priority flow control enablement, NP port enablement, and slow-drain policy values for any interface that this policy group is applied to.
Step 11	Create an interface profile for F port interfaces. Example: <pre>apicl# configure apicl(config)# leaf-interface-profile lip1 apicl(config-leaf-if-profile)# description 'test description lip1' apicl(config-leaf-if-profile)# leaf-interface-group lig1 apicl(config-leaf-if-group)# description 'test description lig1' apicl(config-leaf-if-group)# policy-group ipg1 apicl(config-leaf-if-group)# interface ethernet 1/2-6, 1/9-13</pre>	The sample command sequence creates an interface profile lip1 for F port interfaces, associates the profile with F port specific interface policy group ipg1 , and specifies the interfaces to which this profile and its associated policies applies.
Step 12	Create an interface profile for NP port interfaces. Example: <pre>apicl# configure apicl(config)# leaf-interface-profile lip2</pre>	The sample command sequence creates an interface profile lip2 for NP port interfaces, associates the profile with NP port specific interface policy group ipg2 , and specifies the

	Command or Action	Purpose
	<pre> apic1(config-leaf-if-profile)# description ' test description lip2' apic1(config-leaf-if-profile)# leaf-interface-group lig2 apic1(config-leaf-if-group)# description ' test description lig2' apic1(config-leaf-if-group)# policy-group ipg2 apic1(config-leaf-if-group)# interface ethernet 1/14 </pre>	interface to which this profile and its associated policies applies.
Step 13	Configure QoS Class Policy for Level 1. Example: <pre> apic1(config)# qos parameters level1 apic1(config-qos)# pause no-drop cos 3 </pre>	The sample command sequence specifies the QoS level of FCoE traffic to which priority flow control policy might be applied and pauses no-drop packet handling for Class of Service level 3.

Configuring FCoE Over FEX Using NX-OS Style CLI

FEX ports are configured as port VSANs.

Step 1 Configure Tenant and VSAN domain:

Example:

```

apic1# configure
apic1(config)# tenant t1
apic1(config-tenant)# vrf context v1
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# bridge-domain b1
apic1(config-tenant-bd)# fc
apic1(config-tenant-bd)# vrf member v1
apic1(config-tenant-bd)# exit
apic1(config-tenant)# application a1
apic1(config-tenant-app)# epg e1
apic1(config-tenant-app-epg)# bridge-domain member b1
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)# exit
apic1(config-tenant)# exit

apic1(config)# vsan-domain dom1
apic1(config-vsan)# vlan 1-100
apic1(config-vsan)# vsan 1-100
apic1(config-vsan)# fcoe vsan 2 vlan 2 loadbalancing src-dst-ox-id
apic1(config-vsan)# fcoe vsan 3 vlan 3 loadbalancing src-dst-ox-id
apic1(config-vsan)# fcoe vsan 5 vlan 5
apic1(config-vsan)# exit

```

Step 2 Associate FEX to an interface:

Example:

```

apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/12
apic1(config-leaf-if)# fex associate 111
apic1(config-leaf-if)# exit

```

Step 3 Configure FCoE over FEX per port, port-channel, and VPC:

Example:

```

apic1(config-leaf)# interface vfc 111/1/2
apic1(config-leaf-if)# vsan-domain member dom1
apic1(config-leaf-if)# switchport vsan 2 tenant t1 application a1 epgr e1
apic1(config-leaf-if)# exit

apic1(config-leaf)# interface vfc-po pc1 fex 111
apic1(config-leaf-if)# vsan-domain member dom1
apic1(config-leaf-if)# switchport vsan 2 tenant t1 application a1 epgr e1
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 111/1/3
apic1(config-leaf-if)# channel-group pc1
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit

apic1(config)# vpc domain explicit 12 leaf 101 102
apic1(config-vpc)# exit
apic1(config)# vpc context leaf 101 102
apic1(config-vpc)# interface vpc vpc1 fex 111 111
apic1(config-vpc-if)# vsan-domain member dom1
apic1(config-vpc-if)# switchport vsan 2 tenant t1 application a1 epgr e1
apic1(config-vpc-if)# exit
apic1(config-vpc)# exit
apic1(config)# leaf 101-102
apic1(config-leaf)# interface ethernet 1/2
apic1(config-leaf-if)# fex associate 111
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 111/1/2
apic1(config-leaf-if)# channel-group vpc1 vpc
apic1(config-leaf-if)# exit

```

Step 4 Verify the configuration with the following command:

Example:

```

apic1(config-vpc)# show vsan-domain detail
vsan-domain : dom1

```

```

vsan : 1-100

```

```

vlan : 1-100

```

Leaf State	Interface	Vsan	Vlan	Vsan-Mode	Port-Mode	Usage	Operational
101	vfc111/1/2	2	2	Native		Tenant: t1 App: a1 Epg: e1	Deployed
101	PC:pc1	5	5	Native		Tenant: t1 App: a1 Epg: e1	Deployed
101	vfc111/1/3	3	3	Native	F	Tenant: t1 App: a1 Epg: e1	Deployed

Verifying FCoE Configuration Using the NX-OS Style CLI

The following **show** command verifies the FCoE configuration on your leaf switch ports.

Use the **show vsan-domain** command to verify FCoE is enabled on the target switch.

The command example confirms FCoE enabled on the listed leaf switches and its FCF connection details.

Example:

```
ifav-isim8-ifc1# show vsan-domain detail
```

```
vsan-domain : iPostfcoeDomPl
```

```
vsan : 1-20 51-52 100-102 104-110 200 1999 3100-3101 3133
      2000
```

```
vlan : 1-20 51-52 100-102 104-110 200 1999 3100-3101 3133
      2000
```

Leaf	Interface	Vsan	Vlan	Vsan Mode	Port Mode	Usage	Operational State
101	vfc1/11	1	1	Regular	F	Tenant: iPost101 App: iPost1 Epg: iPost1	Deployed
101	vfc1/12	1	1	Regular	NP	Tenant: iPost101 App: iPost1 Epg: iPost1	Deployed
101	PC:infraAccBndl Grp_pc01	4	4	Regular	NP	Tenant: iPost101 App: iPost4 Epg: iPost4	Deployed
101	vfc1/30	2000		Native		Tenant: t1 App: a1 Epg: e1	Not deployed (invalid-path)

Undeploying FCoE Elements Using the NX-OS Style CLI

Any move to undeploy FCoE connectivity from the ACI fabric requires that you remove the FCoE components on several levels.

Step 1 List the attributes of the leaf port interface, set its mode setting to default, and then remove its EPG deployment and domain association.

The example sets the port mode setting of interface vfc 1/2 to default and then removes the deployment of EPG e1 and the association with VSAN Domain dom1 from that interface.

Example:

```

apicl(config)# leaf 101
apicl(config-leaf)# interface vfc 1/2
apicl(config-leaf-if)# show run
# Command: show running-config leaf 101 interface vfc 1 / 2
# Time: Tue Jul 26 09:41:11 2016
leaf 101
  interface vfc 1/2
    vsan-domain member dom1
    switchport vsan 2 tenant t1 application a1 epq e1
  exit
exit
apicl(config-leaf-if)# no switchport mode
apicl(config-leaf-if)# no switchport vsan 2 tenant t1 application a1 epq e1
apicl(config-leaf-if)# no vsan-domain member dom1
apicl(config-leaf-if)# exit
apicl(config-leaf)# exit

```

Step 2 List and remove the VSAN/VLAN mapping and the VLAN and VSAN pools.

The example removes the VSAN/VLAN mapping for **vsan 2**, VLAN pool **1-10**, and VSAN pool **1-10** from VSAN domain **dom1**.

Example:

```

apicl(config)# vsan-domain dom1
apicl(config-vsan)# show run
# Command: show running-config vsan-domain dom1
# Time: Tue Jul 26 09:43:47 2016
vsan-domain dom1
  vsan 1-10
  vlan 1-10
  fcoe vsan 2 vlan 2
exit
apicl(config-vsan)# no fcoe vsan 2
apicl(config-vsan)# no vlan 1-10
apicl(config-vsan)# no vsan 1-10
apicl(config-vsan)# exit

#####
NOTE: To remove a template-based VSAN to VLAN mapping use an alternate sequence:
#####

apicl(config)# template vsan-attribute <template_name>
apicl(config-vsan-attr)# no fcoe vsan 2

```

Step 3 Delete the VSAN Domain.

The example deletes VSAN domain **dom1**.

Example:

```

apicl(config)# no vsan-domain dom1

```

Step 4 You can delete the associated tenant, EPG, and selectors if you do not need them.

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:

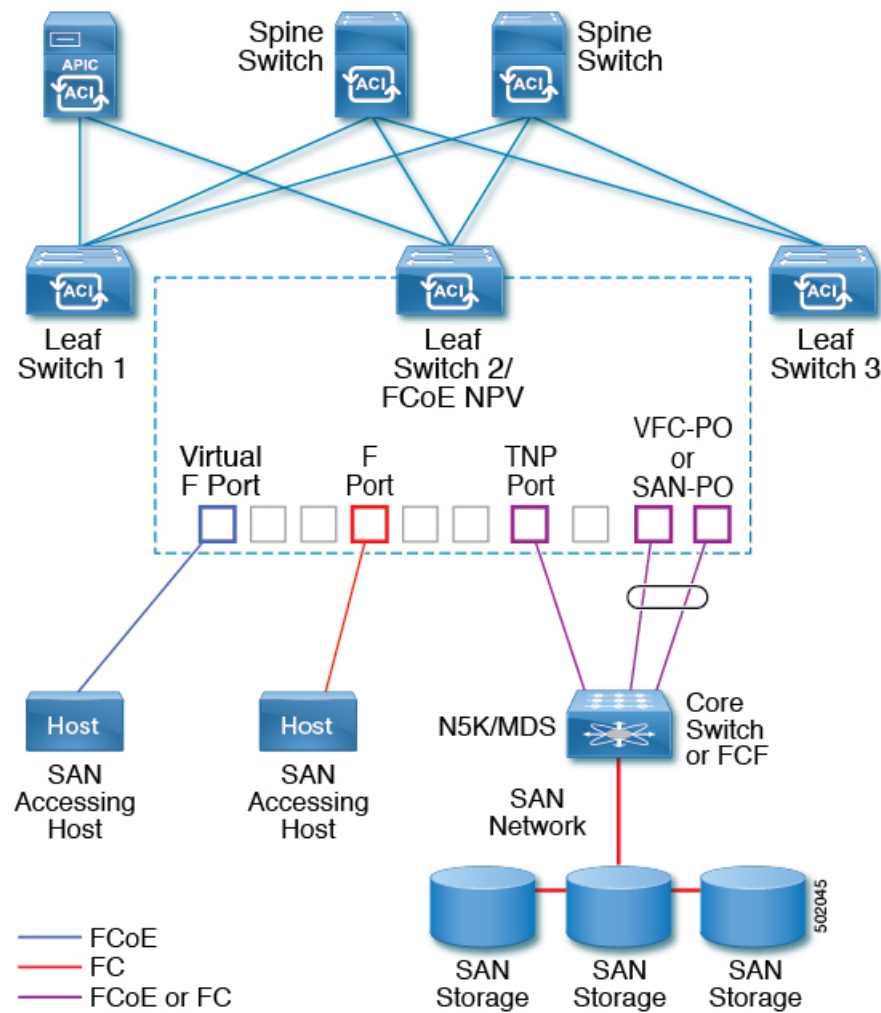
- Increases the number of hosts that connect to the fabric without adding domain IDs in the fabric. The domain ID of the NPV core switch is shared among multiple NPV switches.
- FC and FCoE hosts connect to SAN fabrics using native FC interfaces.
- Automatic traffic mapping for load balancing. For newly 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.
- SAN boot is supported for hosts directly connected by FC F ports. SAN boot is supported on FEX through an FCoE uplink, but not through a vPC.

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.
- FCoE host via FEX over Fibre Channel NPV link is not supported.
- 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.

Configuring FC Connectivity Without Policies or Profiles Using the NX-OS CLI

The sample command sequence below creates bridge domain b1 under tenant t1 configured to support FCoE connectivity.

Before you begin

- Under the target tenant configure a bridge domain to support FCoE traffic.

Step 1 To create a bridge domain for FCoE connectivity:

Example:

```
apic1(config)# tenant t1
apic1(config-tenant)# vrf context v1
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# bridge-domain b1
apic1(config-tenant-bd)# fc
apic1(config-tenant-bd)# vrf member v1
apic1(config-tenant-bd)# exit
apic1(config-tenant)# exit
```

Step 2 Under the same tenant, associate the target EPG with the FCoE-configured bridge domain. The sample command sequence below creates EPG e1 and associates that EPG with the FCoE-configured bridge domain b1:

Example:

```
apic1(config)# tenant t1
apic1(config-tenant)# application a1
apic1(config-tenant-app)# epg e1
apic1(config-tenant-app-epg)# bridge-domain member b1
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)# exit
apic1(config-tenant)# exit
```

Step 3 The following example creates vsan domain dom1 with vsans 1-10:

Example:

```
apic1(config)# vsan-domain dom1
apic1(config-vsan)# vsan 1-10
```

Step 4 Convert range of ports from Ethernet to FC mode. The following example converts port 1/1-4 on switch 101 to FC:

Example:

```
apic1# config
apic1(config)# leaf 101
apic1(config-leaf)# slot 1
apic1(config-leaf-slot)# port 1 4 type fc
apic1(config-leaf-slot)# exit
apic1(config-leaf)# exit
```

Note Port conversion from Ethernet to FC and vice versa requires reload of switch.

- Step 5** Configure FC interface in NP mode. The following example sets various interface properties on interface fc 1/10 and associates that interface with VSAN domain dom1. Each of the targeted interfaces must be assigned one (and only one) VSAN in native mode. The sample command sequence associates the target interface 1/10 with VSAN 10 as a native VSAN and associates it with EPG e1 and application a1 under tenant t1.

Example:

```
apic1(config-leaf)# interface fc 1/10
apic1(config-leaf-fc-if)# switchport mode [f | np]
apic1(config-leaf-fc-if)# switchport rxbbcredit <16-64>
apic1(config-leaf-fc-if)# switchport speed [16G | 32G | 4G | 8G | auto | unknown]
apic1(config-leaf-fc-if)# vsan-domain member dom1
apic1(config-leaf-fc-if)# switchport vsan 10 tenant t1 application a1 epg e1
```

- Step 6** Create traffic map to pin server ports to uplink ports. The following example creates Traffic map for vFC 1/47 server interface pinned to FC 1/7 uplink interface:

Example:

```
apic1# config
apic1(config)# leaf 101
apic1(config-leaf)# npv traffic-map server-interface vfc 1/47 label label1 tenant tenant1 application
  appl epg epg1
apic1(config-leaf)# npv traffic-map external-interface fc 1/7 tenant tenant1 label label1
```

Configuring FC Connectivity With Policies or Profiles Using the NX-OS CLI

The sample command sequence below creates bridge domain b1 under tenant t1 configured to support FCoE connectivity.

Before you begin

- Under the target tenant configure a bridge domain to support FCoE traffic.

- Step 1** To create a bridge domain for FCoE connectivity:

Example:

```
apic1(config)# tenant t1
apic1(config-tenant)# vrf context v1
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# bridge-domain b1
apic1(config-tenant-bd)# fc
apic1(config-tenant-bd)# vrf member v1
apic1(config-tenant-bd)# exit
apic1(config-tenant)# exit
```


- Step 2** Under the same tenant, associate the target EPG with the FCoE-configured bridge domain. The sample command sequence below creates EPG e1 and associates that EPG with the FCoE-configured bridge domain b1:

Example:

```
apic1(config)# tenant t1
apic1(config-tenant)# application a1
apic1(config-tenant-app)# epg e1
apic1(config-tenant-app-epg)# bridge-domain member b1
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)# exit
apic1(config-tenant)# exit
```

- Step 3** Create a VSAN domain. The following example creates vsan domain dom1 with vsans 1-10:

Example:

```
apic1(config)# vsan-domain dom1
apic1(config-vsan)# vsan 1-10
```

- Step 4** Create an interface policy group for NP port interfaces. The sample command sequence creates FC interface policy group ipg2 and assigns a combination of values that determine values for any interface that this policy group is applied to:

Example:

```
apic1(config)# template fc-policy-group ipg1
apic1(config-fc-pol-grp-if)# switchport ?
  fill-pattern  Configure fill pattern for fc interface
  mode          Configure port mode for fc interface
  rxbbcredit    Configure rxBBCredit for fc interface
  speed         Configure speed for fc interface
  trunk-mode    Configure trunk-mode for fc interface
apic1(config-fc-pol-grp-if)# switchport fill-pattern [ARBFF | IDLE]
apic1(config-fc-pol-grp-if)# switchport mode [f | np]
apic1(config-fc-pol-grp-if)# switchport rxbbcredit <16-64>
apic1(config-fc-pol-grp-if)# switchport speed [16G | 32G | 4G | 8G | auto | unknown]
apic1(config-fc-pol-grp-if)# vsan-domain member dom1
```

- Step 5** Create an interface profile for FC port interfaces. The sample command sequence creates an interface profile lip1 for FC port interfaces, associates the profile with FC interface policy group ipg1, and specifies the interfaces to which this profile and its associated policies applies:

Example:

```
apic1# configure
apic1(config)# leaf-interface-profile lip1
apic1(config-leaf-if-profile)# description 'test description lip1'
apic1(config-leaf-if-profile)# leaf-interface-group lig1
apic1(config-leaf-if-group)# description 'test description lig1'
apic1(config-leaf-if-group)# fc-policy-group ipg1
apic1(config-leaf-if-group)# interface fc 1/1-4
```

- Step 6** Create a leaf profile, assign the leaf interface profile to the leaf profile, and assign the leaf IDs on which the profile will be applied:

Example:

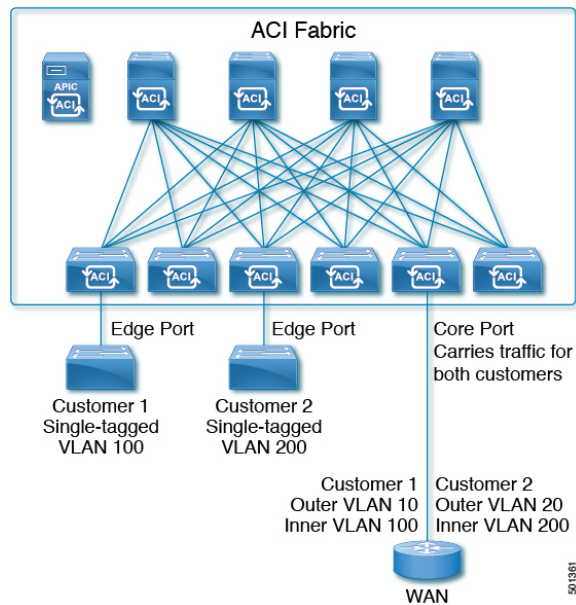
```
apic1(config)# leaf-profile lp103
apic1(config-leaf-profile)# leaf-interface-profile lip1
apic1(config-leaf-profile)# leaf-group range
apic1(config-leaf-group)# leaf 103
apic1(config-leaf-group)#
```

Note After associating leaf interface profile to leaf, reload of leaf is required to bring up these ports as FC ports.

Configuring 802.1Q Tunnels

About ACI 802.1Q Tunnels

Figure 6: 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" on the end of the switch model name.

Terms used in this document may be different in the **Cisco Nexus 9000 Series** documents.

Table 2: 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.
- With Cisco APIC Release 2.2(x) the Ethertypes for double-tagged frames must be 0x9100 followed by 0x8100.
 However, with Cisco APIC Release 2.3(x) and higher, this limitation no longer applies for edge ports, on third-generation Cisco Nexus switches with "FX" on the end of the switch model name.
- 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 Using the NX-OS Style CLI



Note

You can use ports, port-channels, or virtual port channels for interfaces included in a **Dot1q Tunnel**. Detailed steps are included for configuring ports. See the examples below for the commands to configure edge and core port-channels and virtual port channels.

Create a **Dot1q Tunnel** and configure the interfaces for use in the tunnel using the NX-OS Style CLI, with the following steps:



Note

Dot1q Tunnels must include 2 or more interfaces. Repeat the steps (or configure two interfaces together), to mark each interface for use in a **Dot1q Tunnel**. In this example, two interfaces are configured as edge-switch ports, used by a single customer.

Use the following steps to configure a **Dot1q Tunnel** using the NX-OS style CLI:

1. Configure at least two interfaces for use in the tunnel.
2. Create a **Dot1q Tunnel**.
3. Associate all the interfaces with the tunnel.

Before you begin

Configure the tenant that will use the **Dot1q Tunnel**.

SUMMARY STEPS

1. **configure**
2. Configure two interfaces for use in an 802.1Q tunnel, with the following steps:
3. **leaf ID**
4. **interface ethernet slot/port**
5. **switchport mode dot1q-tunnel {edgePort | corePort}**
6. Create an 802.1Q tunnel with the following steps:
7. **leaf ID**
8. **interface ethernet slot/port**
9. **switchport tenant tenant-namedot1q-tunnel tunnel-name**
10. Repeat steps 7 to 10 to associate other interfaces with the tunnel.

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters configuration mode.
Step 2	Configure two interfaces for use in an 802.1Q tunnel, with the following steps:	
Step 3	leaf ID Example: apicl(config)# leaf 101	Identifies the leaf where the interfaces of the Dot1q Tunnel will be located.
Step 4	interface ethernet slot/port Example: apicl(config-leaf)# interface ethernet 1/13-14	Identifies the interface or interfaces to be marked as ports in a tunnel.
Step 5	switchport mode dot1q-tunnel {edgePort corePort} Example: apicl(config-leaf-if)# switchport mode dot1q-tunnel edgePort apicl(config-leaf-if)# exit apicl(config-leaf)# exit apicl(config)# exit	Marks the interfaces for use in an 802.1Q tunnel, and then leaves the configuration mode. The example shows configuring some interfaces for edge port use. Repeat steps 3 to 5 to configure more interfaces for the tunnel.
Step 6	Create an 802.1Q tunnel with the following steps:	
Step 7	leaf ID Example: apicl(config)# leaf 101	Returns to the leaf where the interfaces are located.
Step 8	interface ethernet slot/port Example: apicl(config-leaf)# interface ethernet 1/13-14	Returns to the interfaces included in the tunnel.
Step 9	switchport tenant tenant-namedot1q-tunnel tunnel-name Example: apicl(config-leaf-if)# switchport tenant tenant64 dot1q-tunnel vrf64_edgetunnel apicl(config-leaf-if)# exit	Associates the interfaces to the tunnel and exits the configuration mode.
Step 10	Repeat steps 7 to 10 to associate other interfaces with the tunnel.	

Example: Configuring an 802.1Q Tunnel Using Ports with the NX-OS Style CLI

The example marks two ports as edge port interfaces to be used in a **Dot1q Tunnel**, marks two more ports to be used as core port interfaces, creates the tunnel, and associates the ports with the tunnel.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/13-14
apic1(config-leaf-if)# switchport mode dot1q-tunnel edgePort
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 102
apic1(config-leaf)# interface ethernet 1/10, 1/21
apic1(config-leaf-if)# switchport mode dot1q-tunnel corePort
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# tenant tenant64
apic1(config-tenant)# dot1q-tunnel vrf64_tunnel
apic1(config-tenant-tunnel)# l2protocol-tunnel cdp
apic1(config-tenant-tunnel)# l2protocol-tunnel lldp
apic1(config-tenant-tunnel)# access-encap 200
apic1(config-tenant-tunnel)# mac-learning disable
apic1(config-tenant-tunnel)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/13-14
apic1(config-leaf-if)# switchport tenant tenant64 dot1q-tunnel vrf64_tunnel
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 102
apic1(config-leaf)# interface ethernet 1/10, 1/21
apic1(config-leaf-if)# switchport tenant tenant64 dot1q-tunnel vrf64_tunnel
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

Example: Configuring an 802.1Q Tunnel Using Port-Channels with the NX-OS Style CLI

The example marks two port-channels as edge-port 802.1Q interfaces, marks two more port-channels as core-port 802.1Q interfaces, creates a **Dot1q Tunnel**, and associates the port-channels with the tunnel.

```
apic1# configure
apic1(config)# tenant tenant64
apic1(config-tenant)# dot1q-tunnel vrf64_tunnel
apic1(config-tenant-tunnel)# l2protocol-tunnel cdp
apic1(config-tenant-tunnel)# l2protocol-tunnel lldp
apic1(config-tenant-tunnel)# access-encap 200
apic1(config-tenant-tunnel)# mac-learning disable
apic1(config-tenant-tunnel)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface port-channel pc1
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 1/2-3
apic1(config-leaf-if)# channel-group pc1
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface port-channel pc1
apic1(config-leaf-if)# switchport mode dot1q-tunnel edgePort
apic1(config-leaf-if)# switchport tenant tenant64 dot1q-tunnel vrf64_tunnel
```

```

apic1(config-tenant-tunnel)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 102
apic1(config-leaf)# interface port-channel pc2
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 1/4-5
apic1(config-leaf-if)# channel-group pc2
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface port-channel pc2
apic1(config-leaf-if)# switchport mode dot1q-tunnel corePort
apic1(config-leaf-if)# switchport tenant tenant64 dot1q-tunnel vrf64_tunnel

```

Example: Configuring an 802.1Q Tunnel Using Virtual Port-Channels with the NX-OS Style CLI

The example marks two virtual port-channels (VPCs) as edge-port 802.1Q interfaces for the **Dot1q Tunnel**, marks two more VPCs as core-port interfaces for the tunnel, creates the tunnel, and associates the virtual port-channels with the tunnel.

```

apic1# configure
apic1(config)# vpc domain explicit 1 leaf 101 102
apic1(config)# vpc context leaf 101 102
apic1(config-vpc)# interface vpc vpc1
apic1(config-vpc-if)# switchport mode dot1q-tunnel edgePort
apic1(config-vpc-if)# exit
apic1(config-vpc)# exit
apic1(config)# vpc domain explicit 1 leaf 103 104
apic1(config)# vpc context leaf 103 104
apic1(config-vpc)# interface vpc vpc2
apic1(config-vpc-if)# switchport mode dot1q-tunnel corePort
apic1(config-vpc-if)# exit
apic1(config-vpc)# exit
apic1(config)# tenant tenant64
apic1(config-tenant)# dot1q-tunnel vrf64_tunnel
apic1(config-tenant-tunnel)# l2protocol-tunnel cdp
apic1(config-tenant-tunnel)# l2protocol-tunnel lldp
apic1(config-tenant-tunnel)# access-encap 200
apic1(config-tenant-tunnel)# mac-learning disable
apic1(config-tenant-tunnel)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 103
apic1(config-leaf)# interface ethernet 1/6
apic1(config-leaf-if)# channel-group vpc1 vpc
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 104
apic1(config-leaf)# interface ethernet 1/6
apic1(config-leaf-if)# channel-group vpc1 vpc
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config-vpc)# interface vpc vpc1
apic1(config-vpc-if)# switchport tenant tenant64 dot1q-tunnel vrf64_tunnel
apic1(config-vpc-if)# exit

```

Configuring Dynamic 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

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-FX
- N9K-C9336C-FX2

The 100Gb to 25Gb breakout feature is supported on the access facing ports of the following switches:

- N9K-C93180LC-EX
- N9K-C9336C-FX2

Observe the following guidelines and restrictions:

- In general, breakouts and port profiles (ports changed from uplink to downlink) are not supported on the same port.
- 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 (expect 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.

Configuring Dynamic Breakout Ports Using the NX-OS Style CLI

Use the following steps to configure a breakout port, verify the configuration, and configure an EPG on a sub port, using the NX-OS style CLI.

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.

SUMMARY STEPS

1. **configure**
2. **leaf** *ID*
3. **interface** *ethernetslot/port*
4. **breakout***10g-4x | 25g-4x*
5. **show run**
6. **tenant** *tenant-name*
7. **vrf context** *vrf-name*
8. **bridge-domain** *bridge-domain-name*
9. **vrf member** *vrf-name*
10. **application** *application-profile-name*
11. **epg** *epg-name*
12. **bridge-domain member** *bridge-domain-name*
13. **leaf** *leaf-name*
14. **speed** *interface-speed*
15. **show run**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	leaf <i>ID</i> Example: <code>apic1(config)# leaf 101</code>	Selects the leaf switch where the breakout port will be located and enters leaf configuration mode.
Step 3	interface <i>ethernetslot/port</i> Example: <code>apic1(config-leaf)# interface ethernet 1/16</code>	Identifies the interface to be enabled as a 40 Gigabit Ethernet (GE) breakout port.
Step 4	breakout <i>10g-4x 25g-4x</i> Example: <code>apic1(config-leaf-if)# breakout 10g-4x</code>	Enables the selected interface for breakout. Note For switch support for the Dynamic Breakout Port feature, see Configuration of Dynamic Breakout Ports, on page 116 .

	Command or Action	Purpose
Step 5	show run Example: <pre>apic1(config-leaf-if)# show run # Command: show running-config leaf 101 interface ethernet 1 / 16 # Time: Fri Dec 2 18:13:39 2016 leaf 101 interface ethernet 1/16 breakout 10g-4x apic1(config-leaf-if)# exit apic1(config-leaf)# exit</pre>	Verifies the configuration by showing the running configuration of the interface and returns to global configuration mode.
Step 6	tenant <i>tenant-name</i> Example: <pre>apic1(config)# tenant tenant64</pre>	Selects or creates the tenant that will consume the breakout ports and enters tenant configuration mode.
Step 7	vrf context <i>vrf-name</i> Example: <pre>apic1(config-tenant)# vrf context vrf64 apic1(config-tenant-vrf)# exit</pre>	Creates or identifies the Virtual Routing and Forwarding (VRF) instance associated with the tenant and exits the configuration mode.
Step 8	bridge-domain <i>bridge-domain-name</i> Example: <pre>apic1(config-tenant)# bridge-domain bd64</pre>	Creates or identifies the bridge-domain associated with the tenant and enters BD configuration mode.
Step 9	vrf member <i>vrf-name</i> Example: <pre>apic1(config-tenant-bd)# vrf member vrf64 apic1(config-tenant-bd)# exit</pre>	Associates the VRF with the bridge-domain and exits the configuration mode.
Step 10	application <i>application-profile-name</i> Example: <pre>apic1(config-tenant)# application app64</pre>	Creates or identifies the application profile associated with the tenant and the EPG.
Step 11	epg <i>epg-name</i> Example: <pre>apic1(config-tenant)# epg epg64</pre>	Creates or identifies the EPG and enters into EPG configuration mode.
Step 12	bridge-domain member <i>bridge-domain-name</i> Example: <pre>apic1(config-tenant-app-epg)# bridge-domain member bd64 apic1(config-tenant-app-epg)# exit apic1(config-tenant-app)# exit apic1(config-tenant)# exit</pre>	Associates the EPG with the bridge domain and returns to global configuration mode. Configure the sub ports as desired, for example, use the speed command in leaf interface mode to configure a sub port.
Step 13	leaf <i>leaf-name</i> Example:	Associates the EPG with a break-out port.

	Command or Action	Purpose
	<pre>apic1(config)# leaf 1017 apic1(config-leaf)# interface ethernet 1/13 apic1(config-leaf-if)# vlan-domain member dom1 apic1(config-leaf-if)# switchport trunk allowed vlan 20 tenant t1 application AP1 epg EPG1</pre> Note The vlan-domain and vlan-domain member commands mentioned in the above example are a pre-requisite for deploying an EPG on a port.	
Step 14	speed <i>interface-speed</i> Example: <pre>apic1(config)# leaf 101 apic1(config-leaf)# interface ethernet 1/16/1 apic1(config-leaf-if)# speed 10G apic1(config-leaf-if)# exit</pre>	Enters leaf interface mode, sets the speed of an interface, and exits the configuration mode.
Step 15	show run Example: <pre>apic1(config-leaf)# show run</pre>	After you have configured the sub ports, entering this command in leaf configuration mode displays the sub port details.

The port on leaf 101 at interface 1/16 is confirmed enabled for breakout with sub ports 1/16/1, 1/16/2, 1/16/3, and 1/16/4.

Example

This example configures the port for breakout:

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/16
apic1(config-leaf-if)# breakout 10g-4x
```

This example configures the EPG for the sub ports.

```
apic1(config)# tenant tenant64
apic1(config-tenant)# vrf context vrf64
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# bridge-domain bd64
apic1(config-tenant-bd)# vrf member vrf64
apic1(config-tenant-bd)# exit
apic1(config-tenant)# application app64
apic1(config-tenant-app)# epg epg64
apic1(config-tenant-app-epg)# bridge-domain member bd64
apic1(config-tenant-app-epg)# end
```

This example sets the speed for the breakout sub ports to 10G.

```
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/16/1
apic1(config-leaf-if)# speed 10G
apic1(config-leaf-if)# exit

apic1(config-leaf)# interface ethernet 1/16/2
apic1(config-leaf-if)# speed 10G
```

```

apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 1/16/3
apic1(config-leaf-if)# speed 10G
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 1/16/4
apic1(config-leaf-if)# speed 10G
apic1(config-leaf-if)# exit

```

This example shows the four sub ports connected to leaf 101, interface 1/16.

```

apic1#(config-leaf)# show run
# Command: show running-config leaf 101
# Time: Fri Dec 2 00:51:08 2016
leaf 101
  interface ethernet 1/16/1
    speed 10G
    negotiate auto
    link debounce time 100
    exit
  interface ethernet 1/16/2
    speed 10G
    negotiate auto
    link debounce time 100
    exit
  interface ethernet 1/16/3
    speed 10G
    negotiate auto
    link debounce time 100
    exit
  interface ethernet 1/16/4
    speed 10G
    negotiate auto
    link debounce time 100
    exit
  interface ethernet 1/16
    breakout 10g-4x
    exit
  interface vfc 1/16

```

Configuring Port Profiles

Configuring Port Profiles

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 (only uplink to downlink conversion supported)
- N9K-C93180YC-EX, and N9K-C93180YC-EXU
- N9K-C93108TC-EX and N9K-C93108TC-FX

- N9K-C9336C-FX2 (ports 31-34 can be converted to downlink)
- 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.

Guidelines

In converting uplinks to downlinks and downlinks to uplinks, consider the following guidelines.

Subject	Guideline
Decommissioning nodes with port profiles	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 -k</code> script, and wait for it to run. Then, enter the <code>reload</code> command. The <code>-k</code> script option allows the port-profile setting to persist across the reload, making an additional reboot unnecessary.
FIPS	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. FIPS is supported on Cisco NX-OS release 13.1(1) or later. 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.

Subject	Guideline
Maximum uplink port limit	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: 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). 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. 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.

Breakout Limitations

Switch	Releases	Limitations
N9K-C9332PQ	Cisco APIC 2.2 (1n) and higher	• 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	• 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	• 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	• 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: • Convert ports 49-52 to front panel ports (downlinks). • Perform a port-profile reload, using one of the following methods: • In the APIC GUI, navigate to Fabric > Inventory > Pod > Leaf, right-click Chassis and choose Reload. • In the NX-OS style CLI, enter the setup-clean-config.sh -k script, wait for it to run, and then 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)
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	24 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 NX-OS Style CLI

To configure a port profile using the NX-OS style CLI, perform the following steps:

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 **configure**

Enters global configuration mode.

Example:

```
apic1# configure
```

Step 2 **leaf *node-id***

Specifies the leaf or leaf switches to be configured.

Example:

```
apic1(config)# leaf 102
```

Step 3 **interface *type***

Specifies the interface that you are configuring. You can specify the interface type and identity. For an Ethernet port, use *ethernet slot / port*.

Example:

```
apic1(config-leaf)# interface ethernet 1/2
```

Step 4 **port-direction {uplink | downlink}**

Determines the port direction or changes it. This example configures the port to be a downlink.

Note On the N9K-C9336C-FX switch, changing a port from uplink to downlink is not supported.

Example:

```
apic1(config-leaf-if)# port-direction downlink
```

Step 5 Log on to the leaf switch where the port is located and enter the **setup-clean-config.sh -k** command, then the **reload** command.

Verifying Port Profile Configuration and Conversion Using the NX-OS Style CLI

You can verify the configuration and the conversion of the ports using the **show interface brief** CLI command.



Note Port profile can be deployed only on the top ports of a Cisco N9K-C93180LC-EX switch, for example, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, and 23. When the top port is converted using the port profile, the bottom ports are hardware disabled. For example, if Eth 1/1 is converted using the port profile, Eth 1/2 is hardware disabled.

Step 1 This example displays the output for converting an uplink port to downlink port. Before converting an uplink port to downlink port, the output is displayed in the example. The keyword **routed** denotes the port as uplink port.

Example:

```
switch# show interface brief
<snip>
Eth1/49          --      eth  routed  down  sfp-missing          100G (D)  --
Eth1/50          --      eth  routed  down  sfp-missing          100G (D)  --
<snip>
```

Step 2 After configuring the port profile and reloading the switch, the output is displayed in the example. The keyword **trunk** denotes the port as downlink port.

Example:

```
switch# show interface brief
<snip>
Eth1/49          0        eth  trunk   down  sfp-missing          100G (D)  --
Eth1/50          0        eth  trunk   down  sfp-missing          100G (D)  --
<snip>
```

Microsegmentation on Virtual Switches

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 NX-OS-Style CLI

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 VM-based attributes within an application EPG.

Step 1 In the CLI, enter configuration mode:

Example:

```
apic1# configure
apic1(config)#
```

Step 2 Create the uSeg EPG:

Example:

This example is for an application EPG.

Note The command to allow microsegmentation in the following example is required for VMware VDS only.

```
apic1(config)# tenant cli-ten1
apic1(config-tenant)# application cli-a1
apic1(config-tenant-app)# epg cli-baseEPG1
apic1(config-tenant-app-epg)# bridge-domain member cli-bd1
apic1(config-tenant-app-epg)# vmware-domain member cli-vmml allow-micro-segmentation
```

Example:

(Optional) This example sets match EPG precedence for the uSeg EPG:

```
apic1(config)# tenant Coke
apic1(config-tenant)# application cli-a1
apic1(config-tenant-app)# epg cli-uepg1 type micro-segmented
apic1(config-tenant-app-uepg)# bridge-domain member cli-bd1
apic1(config-tenant-app-uepg)# match-precedence 10
```

Example:

This example uses a filter based on the attribute VM Name.

```
apic1(config)# tenant cli-ten1
apic1(config-tenant)# application cli-a1
apic1(config-tenant-app)# epg cli-uepg1 type micro-segmented
apic1(config-tenant-app-uepg)# bridge-domain member cli-bd1
apic1(config-tenant-app-uepg)# attribute-logical-expression 'vm-name contains <cos1>'
```

Example:

This example uses a filter based on an IP address.

```
apic1(config)# tenant cli-ten1
apic1(config-tenant)# application cli-a1
apic1(config-tenant-app)# epg cli-uepg1 type micro-segmented
apic1(config-tenant-app-uepg)# bridge-domain member cli-bd1
apic1(config-tenant-app-uepg)# attribute-logical-expression 'ip equals <FF:FF:FF:FF:FF:FF>'
```

Example:

This example uses a filter based on a MAC address.

```
apic1(config)# tenant cli-ten1
apic1(config-tenant)# application cli-a1
apic1(config-tenant-app)# epg cli-uepg1 type micro-segmented
apic1(config-tenant-app-uepg)# bridge-domain member cli-bd1
apic1(config-tenant-app-uepg)# attribute-logical-expression 'mac equals <FF-FF-FF-FF-FF-FF>'
```

Example:

This example uses the operator AND to match all attributes and the operator OR to match any attribute.

```
apic1(config)# tenant cli-ten1
apic1(config-tenant)# application cli-a1
apic1(config-tenant-app)# epg cli-uepg1 type micro-segmented
apic1(config-tenant-app-uepg)# attribute-logical-expression 'hv equals host-123 OR (guest-os equals
  "Ubuntu Linux (64-bit)" AND domain contains fex)'
```

Example:

This example uses a filter based on the attribute VM-Custom Attribute.

```
apic1(config)# tenant cli-ten1
apic1(config-tenant)# application cli-a1
apic1(config-tenant-app)# epg cli-uepg1 type micro-segmented
apic1(config-tenant-app-uepg)# bridge-domain member cli-bd1
apic1(config-tenant-app-uepg)# attribute-logical-expression 'custom <Custom Attribute Name> equals
  <Custom Attribute value>'
```

Step 3 (Cisco ACI Virtual Edge only): Attach the uSeg EPG to a Cisco ACI Virtual Edge VMM domain, specifying the switching and encapsulation modes:

Example:

```
vmware-domain member AVE-CISCO
  switching-mode AVE
  encap-mode vxlan
  exit
```

Step 4 Verify the uSeg EPG creation:

Example:

The following example is for a uSeg EPG with a VM name attribute filter

```
apic1(config-tenant-app-uepg)# show running-config
# Command: show running-config tenant cli-ten1 application cli-a1 epg cli-uepg1 type micro-segmented
# Time: Thu Oct 8 11:54:32 2015
tenant cli-ten1
  application cli-a1
    epg cli-uepg1 type micro-segmented
      bridge-domain cli-bd1
      attribute-logical-expression 'vm-name contains cos1 force'
      {vmware-domain | microsoft-domain} member cli-vmm1
    exit
  exit
exit
```

Configuring Microsegmentation on Bare-Metal

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 simply 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 a Network-Based Microsegmented EPG in a Bare-Metal Environment Using the NX-OS Style CLI

This section describes how to configure microsegmentation with Cisco ACI using network-based attributes (IP address or MAC address) within a base EPG in a bare-metal environment.

SUMMARY STEPS

1. In the CLI, enter configuration mode:
2. Create the microsegment:
3. Deploy the EPG.
4. Verify the microsegment creation:

DETAILED STEPS

	Command or Action	Purpose
Step 1	In the CLI, enter configuration mode: Example: apic1# configure apic1(config)#	
Step 2	Create the microsegment: Example: This example uses a filter based on an IP address. apic1(config)# tenant cli-ten1 apic1(config-tenant)# application cli-a1 apic1(config-tenant-app)# epg cli-uepg1 type micro-segmented apic1(config-tenant-app-uepg)# bridge-domain member cli-bd1 apic1(config-tenant-app-uepg)# attribute cli-upg-att match ip <X.X.X.X> #Schemes to express the ip A.B.C.D IP Address A.B.C.D/LEN IP Address and mask Example: This example uses a filter based on a MAC address.	

	Command or Action	Purpose
	<pre> apicl(config)# tenant cli-ten1 apicl(config-tenant)# application cli-a1 apicl(config-tenant-app)# epg cli-uepg1 type micro-segmented apicl(config-tenant-app-uepg)# bridge-domain member cli-bd1 apicl(config-tenant-app-uepg)# attribute cli-upg-att match mac <FF-FF-FF-FF-FF-FF> #Schemes to express the mac E.E.E MAC address (Option 1) EE-EE-EE-EE-EE-EE MAC address (Option 2) EE:EE:EE:EE:EE:EE MAC address (Option 3) EEEE.EEEE.EEEE MAC address (Option 4) </pre> Example: This example uses a filter based on a MAC address and enforces intra-EPG isolation between all members of this uSeg EPG: <pre> apicl(config)# tenant cli-ten1 apicl(config-tenant)# application cli-a1 apicl(config-tenant-app)# epg cli-uepg1 type micro-segmented apicl(config-tenant-app-uepg)# isolation enforced apicl(config-tenant-app-uepg)# bridge-domain member cli-bd1 apicl(config-tenant-app-uepg)# attribute cli-upg-att match mac <FF-FF-FF-FF-FF-FF> #Schemes to express the mac E.E.E MAC address (Option 1) EE-EE-EE-EE-EE-EE MAC address (Option 2) EE:EE:EE:EE:EE:EE MAC address (Option 3) EEEE.EEEE.EEEE MAC address (Option 4) </pre>	
Step 3	Deploy the EPG. Example: This example deploys the EPG and bids to the leaf. <pre> apicl(config)# leaf 101 apicl(config-leaf)# deploy-epg tenant cli-ten1 application cli-a1 epg cli-uepg1 type micro-segmented </pre>	
Step 4	Verify the microsegment creation: Example: <pre> apicl(config-tenant-app-uepg)# show running-config # Command: show running-config tenant cli-ten1 application cli-appl epg cli-uepg1 type micro-segmented # Time: Thu Oct 8 11:54:32 2015 tenant cli-ten1 application cli-appl epg cli-esx1bu type micro-segmented bridge-domain cli-bd1 attribute cli-uepg-att match mac 00:11:22:33:44:55 exit </pre>	

	Command or Action	Purpose
	exit exit	

Configuring Layer 2 IGMP Snoop Multicast

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**.

Enabling IGMP Snooping Static Port Groups

IGMP static port grouping enables you to pre-provision ports, that were previously statically-assigned to an application EPG, to enable 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 assigned to an application EPG.

Static group membership can be configured through the APIC GUI, CLI, and REST API interfaces.

Configuring and Assigning an IGMP Snooping Policy to a Bridge Domain using the NX-OS Style CLI

Before you begin

- Create the tenant that will consume the IGMP Snooping policy.
- Create the bridge domain for the tenant, where you will attach the IGMP Snooping policy.

Procedure

	Command or Action	Purpose
Step 1	Create a snooping policy based on default values. Example: <pre> apic1(config-tenant)# template ip igmp snooping policy cookieCut1 apic1(config-tenant-template-ip-igmp-snooping)# show run all </pre> <pre> # Command: show running -config all tenant foo template ip igmp snooping policy cookieCut1 # Time: Thu Oct 13 18:26:03 2016 tenant t_10 template ip igmp snooping policy cookieCut1 ip igmp snooping no ip igmp snooping fast-leave ip igmp snooping last-member-query-interval 1 no ip igmp snooping querier ip igmp snooping query-interval 125 ip igmp snooping query-max-response-time 10 ip igmp snooping stqrtp-up-query-count 2 ip igmp snooping startup-query-interval 31 no description exit exit apic1(config-tenant-template-ip-igmp-snooping)# </pre>	The example NX-OS style CLI sequence: • Creates an IGMP Snooping policy named cookieCut1 with default values. • Displays the default IGMP Snooping values for the policy cookieCut1.
Step 2	Modify the snooping policy as necessary. Example: <pre> apic1(config-tenant-template-ip-igmp-snooping)# ip </pre>	The example NX-OS style CLI sequence: • Specifies a custom value for the query-interval value in the IGMP Snooping policy named cookieCut1.

	Command or Action	Purpose
	<pre> igmp snooping query-interval 300 apic1(config-tenant-template-ip-igmp-snooping)# show run all # Command: show running -config all tenant foo template ip igmp snooping policy cookieCut1 #Time: Thu Oct 13 18:26:03 2016 tenant foo template ip igmp snooping policy cookieCut1 ip igmp snooping no ip igmp snooping fast-leave ip igmp snooping last-member-query-interval 1 no ip igmp snooping querier ip igmp snooping query-interval 300 ip igmp snooping query-max-response-time 10 ip igmp snooping stqrtup-query-count 2 ip igmp snooping startup-query-interval 31 no description exit exit apic1(config-tenant-template-ip-igmp-snooping)# exit apic1(config--tenant)# </pre>	• Confirms the modified IGMP Snooping value for the policy cookieCut1.
Step 3	Assign the policy to a bridge domain. Example: <pre> apic1(config-tenant)# int bridge-domain bd3 apic1(config-tenant-interface)# ip igmp snooping policy cookieCut1 </pre>	The example NX-OS style CLI sequence: • Navigates to bridge domain, BD3. for the query-interval value in the IGMP Snooping policy named cookieCut1. • Assigns the IGMP Snooping policy with a modified IGMP Snooping value for the policy cookieCut1.

What to do next

You can assign the IGMP Snooping policy to multiple bridge domains.

Enabling IGMP Snooping and Multicast on Static Ports in the NX-OS Style CLI

You can enable IGMP snooping and multicast on ports that have been statically assigned to an EPG. Then you can create and assign access groups of users that are permitted or denied access to the IGMP snooping and multicast traffic enabled on those ports.

The steps described in this task assume the pre-configuration of the following entities:

- Tenant: tenant_A
- Application: application_A
- EPG: epG_A
- Bridge Domain: bridge_domain_A
- vrf: vrf_A -- a member of bridge_domain_A
- VLAN Domain: vd_A (configured with a range of 300-310)

- Leaf switch: 101 and interface 1/10

The target interface 1/10 on switch 101 is associated with VLAN 305 and statically linked with tenant_A, application_A, epg_A

- Leaf switch: 101 and interface 1/11

The target interface 1/11 on switch 101 is associated with VLAN 309 and statically linked with tenant_A, application_A, epg_A

Before you begin

Before you begin to enable IGMP snooping and multicasting for an EPG, complete the following tasks.

- Identify the interfaces to enable this function and statically assign them to that EPG



Note

For details on static port assignment, see *Deploying an EPG on a Specific Port with APIC Using the NX-OS Style CLI* in the *Cisco APIC Layer 2 Networking Configuration Guide*.

- Identify the IP addresses that you want to be recipients of IGMP snooping multicast traffic.

Procedure

	Command or Action	Purpose
Step 1	On the target interfaces enable IGMP snooping and layer 2 multicasting Example: <pre> apic1# conf t apic1(config)# tenant tenant_A apic1(config-tenant)# application application_A apic1(config-tenant-app)# epg epg_A apic1(config-tenant-app-epg)# ip igmp snooping static-group 225.1.1.1 leaf 101 interface ethernet 1/10 vlan 305 apic1(config-tenant-app-epg)# end apic1# conf t apic1(config)# tenant tenant_A; application application_A; epg epg_A apic1(config-tenant-app-epg)# ip igmp snooping static-group 227.1.1.1 leaf 101 interface ethernet 1/11 vlan 309 apic1(config-tenant-app-epg)# exit apic1(config-tenant-app)# exit </pre>	The example sequences enable: • IGMP snooping on the statically-linked target interface 1/10 and associates it with a multicast IP address, 225.1.1.1 • IGMP snooping on the statically-linked target interface 1/11 and associates it with a multicast IP address, 227.1.1.1

Enabling IGMP Snoop Access Groups

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.

IGMP snoop access groups can be configured through the APIC GUI, CLI, and REST API interfaces.

Enabling Group Access to IGMP Snooping and Multicast using the NX-OS Style CLI

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.

The steps described in this task assume the pre-configuration of the following entities:

- Tenant: tenant_A
- Application: application_A
- EPG: epg_A
- Bridge Domain: bridge_domain_A
- vrf: vrf_A -- a member of bridge_domain_A
- VLAN Domain: vd_A (configured with a range of 300-310)
- Leaf switch: 101 and interface 1/10

The target interface 1/10 on switch 101 is associated with VLAN 305 and statically linked with tenant_A, application_A, epg_A

- Leaf switch: 101 and interface 1/11

The target interface 1/11 on switch 101 is associated with VLAN 309 and statically linked with tenant_A, application_A, epg_A



Note

For details on static port assignment, see *Deploying an EPG on a Specific Port with APIC Using the NX-OS Style CLI* in the *Cisco APIC Layer 2 Networking Configuration Guide*.

Procedure

	Command or Action	Purpose
Step 1	Define the route-map "access groups." Example: <pre>apic1# conf t apic1(config)# tenant tenant_A; application application_A; epg epg_A apic1(config-tenant)# route-map fooBroker permit apic1(config-tenant-rtmap)# match ip multicast group 225.1.1.1/24 apic1(config-tenant-rtmap)# exit apic1(config-tenant)# route-map fooBroker deny apic1(config-tenant-rtmap)# match ip multicast</pre>	The example sequences configure: • Route-map-access group "foobroker" linked to multicast group 225.1.1.1/24, access permitted • Route-map-access group "foobroker" linked to multicast group 227.1.1.1/24, access denied

	Command or Action	Purpose
	<pre>group 227.1.1.1/24 apicl(config-tenant-rtmap)# exit</pre>	
Step 2	Verify route map configurations. Example: <pre>apicl(config-tenant)# show running-config tenant test route-map fooBroker # Command: show running-config tenant test route-map fooBroker # Time: Mon Aug 29 14:34:30 2016 tenant test route-map fooBroker permit 10 match ip multicast group 225.1.1.1/24 exit route-map fooBroker deny 20 match ip multicast group 227.1.1.1/24 exit exit</pre>	
Step 3	Specify the access group connection path. Example: <pre>apicl(config-tenant)# application application_A apicl(config-tenant-app)# epg epg_A apicl(config-tenant-app-epg)# ip igmp snooping access-group route-map fooBroker leaf 101 interface ethernet 1/10 vlan 305 apicl(config-tenant-app-epg)# ip igmp snooping access-group route-map newBroker leaf 101 interface ethernet 1/10 vlan 305</pre>	The example sequences configure: • Route-map-access group "foobroker" connected through leaf switch 101, interface 1/10, and VLAN 305. • Route-map-access group "newbroker" connected through leaf switch 101, interface 1/10, and VLAN 305.
Step 4	Verify the access group connections. Example: <pre>apicl(config-tenant-app-epg)# show run # Command: show running-config tenant tenant_A application application_A epg epg_A # Time: Mon Aug 29 14:43:02 2016 tenant tenant_A application application_A epg epg_A bridge-domain member bridge_domain_A ip igmp snooping access-group route-map fooBroker leaf 101 interface ethernet 1/10 vlan 305 ip igmp snooping access-group route-map fooBroker leaf 101 interface ethernet 1/11 vlan 309 ip igmp snooping access-group route-map newBroker leaf 101 interface ethernet 1/10 vlan 305 ip igmp snooping static-group 225.1.1.1 leaf 101 interface ethernet 1/10 vlan 305 ip igmp snooping static-group 225.1.1.1 leaf 101 interface ethernet 1/11 vlan 309 exit exit exit</pre>	

Deploying an EPG on a Specific Port with APIC Using the NX-OS Style CLI

Step 1 Configure a VLAN domain:

Example:

```
apic1(config)# vlan-domain dom1
apic1(config-vlan)# vlan 10-100
```

Step 2 Create a tenant:

Example:

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

Step 3 Create a private network/VRF:

Example:

```
apic1(config-tenant)# vrf context ctx1
apic1(config-tenant-vrf)# exit
```

Step 4 Create a bridge domain:

Example:

```
apic1(config-tenant)# bridge-domain bd1
apic1(config-tenant-bd)# vrf member ctx1
apic1(config-tenant-bd)# exit
```

Step 5 Create an application profile and an application EPG:

Example:

```
apic1(config-tenant)# application AP1
apic1(config-tenant-app)# epg EPG1
apic1(config-tenant-app-epg)# bridge-domain member bd1
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)# exit
apic1(config-tenant)# exit
```

Step 6 Associate the EPG with a specific port:

Example:

```
apic1(config)# leaf 1017
apic1(config-leaf)# interface ethernet 1/13
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk allowed vlan 20 tenant t1 application AP1 epg EPG1
```

Note The vlan-domain and vlan-domain member commands mentioned in the above example are a pre-requisite for deploying an EPG on a port.

Configuring 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 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.

Configuring a Port Security Policy Group Template

SUMMARY STEPS

1. **configure**
2. **[no] template policy-group *policy-group-name***
3. **[no] switchport access vlan *vlan-id* tenant *tenant-name* application *application-name* epg *epg-name***
4. **[no] switchport port-security maximum *number-of-addresses***

5. **[no] switchport port-security violation protect**
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	[no] template policy-group <i>policy-group-name</i> Example: <code>apic1(config)# template policy-group PortSecGrp1</code>	Creates (or deletes) a policy group template.
Step 3	[no] switchport access vlan <i>vlan-id</i> tenant <i>tenant-name</i> application <i>application-name</i> epg <i>epg-name</i> Example: <code>apic1(config-pol-grp-if)# switchport access vlan 4 tenant ExampleCorp application Web epg webEpg</code>	
Step 4	[no] switchport port-security maximum <i>number-of-addresses</i> Example: <code>apic1(config-pol-grp-if)# switchport port-security maximum 1</code>	Sets the maximum number of secure MAC addresses for the port. The range is 0 to 12000 addresses. The default is 1 address.
Step 5	[no] switchport port-security violation protect Example: <code>apic1(config-pol-grp-if)# switchport port-security violation protect</code>	Sets the action to be taken when a security violation is detected. The protect action drops packets with unknown source addresses until you remove a sufficient number of secure MAC addresses to drop below the maximum value.
Step 6	exit Example: <code>apic1(config-pol-grp-if)# exit</code>	Returns to global configuration mode.

Example

This example shows how to create a port security policy group template.

```
apic1# configure
apic1(config)# template policy-group PortSecGrp1
apic1(config-pol-grp-if)# switchport port-security maximum 20
apic1(config-pol-grp-if)# switchport port-security violation protect
apic1(config-pol-grp-if)# exit
```

What to do next

Apply the port security template to an interface.

Configuring Port Security on an Interface Using a Template

Before you begin

Create a port security policy group template.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **interface** *type-or-range*
4. **[no] policy-group** *policy-group-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: apicl(config)# leaf 101	Specifies the leaf to be configured.
Step 3	interface <i>type-or-range</i> Example: apicl(config-leaf)# interface eth 1/2-4	Specifies a port or a range of ports to be configure.
Step 4	[no] policy-group <i>policy-group-name</i> Example: apicl(config-leaf-if)# policy-group PortSecGrp1	Applies the policy group template to the port or range of ports.

Example

This example shows how to apply a port security policy group template to a range of Ethernet ports.

```

apicl# configure
apicl(config)# leaf 101
apicl(config-leaf)# interface eth 1/2-4
apicl(config-leaf-if)# policy-group PortSecGrp1

```

This example shows how to configure port security on a port channel using a template.


```

apic1# configure
apic1(config)# template port-channel po1
apic1(config-if)# switchport port-security maximum 10
apic1(config-if)# switchport port-security violation protect
apic1(config-if)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface eth 1/3-4
apic1(config-leaf-if)# channel-group po1
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit

```

Configuring Port Security on an Interface Using Overrides

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **interface** *type-or-range*
4. **[no] switchport port-security maximum** *number-of-addresses*
5. **[no] switchport port-security violation protect**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: apic1(config)# leaf 101	Specifies the leaf to be configured.
Step 3	interface <i>type-or-range</i> Example: apic1(config-leaf)# interface eth 1/2-4	Specifies an interface or a range of interfaces to be configured.
Step 4	[no] switchport port-security maximum <i>number-of-addresses</i> Example: apic1(config-leaf-if)# switchport port-security maximum 1	Sets the maximum number of secure MAC addresses for the interface. The range is 0 to 12000 addresses. The default is 1 address.
Step 5	[no] switchport port-security violation protect Example: apic1(config-leaf-if)# switchport port-security violation protect	Sets the action to be taken when a security violation is detected. The protect action drops packets with unknown source addresses until you remove a sufficient number of secure MAC addresses to drop below the maximum value.

Example

This example shows how to configure port security on an Ethernet interface.

```
apicl# configure
apicl(config)# leaf 101
apicl(config-leaf)# interface eth 1/2
apicl(config-leaf-if)# switchport port-security maximum 10
apicl(config-leaf-if)# switchport port-security violation protect
```

This example shows how to configure port security on a port channel.

```
apicl# configure
apicl(config)# leaf 101
apicl(config-leaf)# interface port-channel po2
apicl(config-leaf-if)# switchport port-security maximum 10
apicl(config-leaf-if)# switchport port-security violation protect
```

This example shows how to configure port security on a virtual port channel (VPC).

```
apicl# configure
apicl(config)# vpc domain explicit 1 leaf 101 102
apicl(config-vpc)# exit
apicl(config)# template port-channel po4
apicl(config-if)# exit
apicl(config)# leaf 101-102
apicl(config-leaf)# interface eth 1/11-12
apicl(config-leaf-if)# channel-group po4 vpc
apicl(config-leaf-if)# exit
apicl(config-leaf)# exit
apicl(config)# vpc context leaf 101 102
apicl(config-vpc)# interface vpc po4
apicl(config-vpc-if)# switchport port-security maximum 10
apicl(config-vpc-if)# switchport port-security violation protect
```

802.1x Port and Node Authentication

802.1x Port and Node Authentication

IEEE 802.1x is a port-based authentication mechanism to prevent unauthorized devices from gaining access to the network. You can configure 802.1x port and node authentication using the NX-OS style CLI.

Configuring a Port Authentication Policy

SUMMARY STEPS

1. In the CLI, enter configuration mode:
2. Create a policy group:
3. Configure port-level authentication policy in the policy group:

4. Configure host mode (two modes are supported: multi-host and single-host - single being the default setting):
5. Enable this policy (policy is disabled by default):
6. Configure the leaf interface profile:
7. Configure a policy group for the leaf switch interface profile:
8. Specify ports and/or interfaces for your interface group:
9. Apply the policy on your interface group:
10. Configure the leaf profile :
11. Configure the leaf policy group and specify leaf switch nodes for the group:
12. Apply an interface policy on the leaf switch profile:

DETAILED STEPS

Step 1 In the CLI, enter configuration mode:

Example:

```
apic1# configure
apic1(config)#
```

Step 2 Create a policy group:

Example:

```
apic1(config)# template policy-group mypol
```

Step 3 Configure port-level authentication policy in the policy group:

Example:

```
apic1(config-pol-grp-if) # switchport port-authentication mydot1x
```

Step 4 Configure host mode (two modes are supported: multi-host and single-host - single being the default setting):

Example:

```
apic1(config-port-authentication) # host-mode multi-host
```

Step 5 Enable this policy (policy is disabled by default):

Example:

```
apic1(config-port-authentication) # no shutdown
apic1(config-port-authentication) # exit
apic1(config-pol-grp-if) # exit
apic1(config)#
```

Step 6 Configure the leaf interface profile:

Example:

```
apic1(config)# leaf-interface-profile myprofile
```

Step 7 Configure a policy group for the leaf switch interface profile:

Example:

```
apic1(config-leaf-if-profile) # leaf-interface-group mygroup
```

Step 8 Specify ports and/or interfaces for your interface group:

Example:

```
apic1(config-leaf-if-group)# interface ethernet 1/10-12
```

Step 9 Apply the policy on your interface group:

Example:

```
apic1(config-leaf-if-group)# policy-group mypol
apic1(config-leaf-if-group)# exit
apic1(config-leaf-if-profile)# exit
```

Step 10 Configure the leaf profile :

Example:

```
apic1(config)#
apic1(config)# leaf-profile myleafprofile
```

Step 11 Configure the leaf policy group and specify leaf switch nodes for the group:

Example:

```
apic1(config-leaf-profile)# leaf-group myleafgrp
apic1(config-leaf-group)# leaf 101
apic1(config-leaf-group)# exit
```

Step 12 Apply an interface policy on the leaf switch profile:

Example:

```
apic1(config-leaf-profile)# leaf-interface-profile myprofile
apic1(config-leaf-group)# exit
apic1(config)#
```

Configuring a Node Authentication Policy

SUMMARY STEPS

1. In the CLI, enter configuration mode:
2. Configure the radius authentication group:
3. Configure node level port authentication policy:
4. [Optional] Override the default VLAN ID if authentication fails. :
5. [Optional] Override default EPG if authentication fails:
6. Configure policy group and specify port authentication policy in the group:
7. Configure the leaf switch profile:
8. Configure a group for the leaf switch profile and specify the policy group:

DETAILED STEPS

Step 1 In the CLI, enter configuration mode:

Example:

```
apic1# configure
apic1(config)#
```

Step 2 Configure the radius authentication group:

Example:

```
apic1(config)# aaa group server radius myradiusgrp
apic1(config-radius)#server 192.168.0.100 priority 1
apic1(config-radius)#exit
```

Step 3 Configure node level port authentication policy:

Example:

```
apic1(config)# policy-map type port-authentication mydot1x
apic1(config-pmap-port-authentication)#radius-provider-group myradiusgrp
```

Step 4 [Optional] Override the default VLAN ID if authentication fails. :

Example:

```
apic1(config-pmap-port-authentication)#fail-auth-vlan 2001
```

Step 5 [Optional] Override default EPG if authentication fails:

Example:

```
apic1(config-pmap-port-authentication)#fail-auth-epg tenant tn1 application ap1 epg epg256
apic1(config)# exit
```

Step 6 Configure policy group and specify port authentication policy in the group:

Example:

```
apic1(config)#template leaf-policy-group lpg2
apic1(config-leaf-policy-group)# port-authentication mydot1x
apic1(config-leaf-policy-group)#exit
```

Step 7 Configure the leaf switch profile:

Example:

```
apic1(config)# leaf-profile mylp2
```

Step 8 Configure a group for the leaf switch profile and specify the policy group:

Example:

```
apic1(config-leaf-profile)#leaf-group mylg2
apic1(config-leaf-group)# leaf-policy-group lpg2
apic1(config-leaf-group)#exit
```

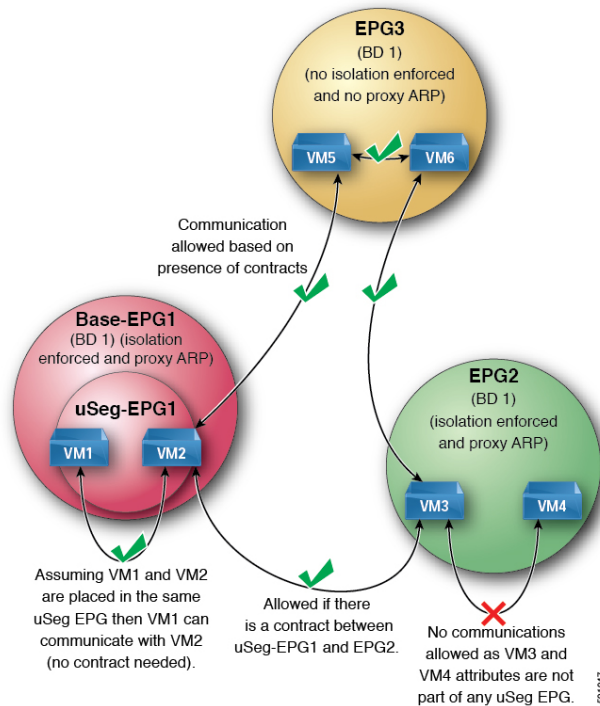
Configuring Proxy ARP

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*.

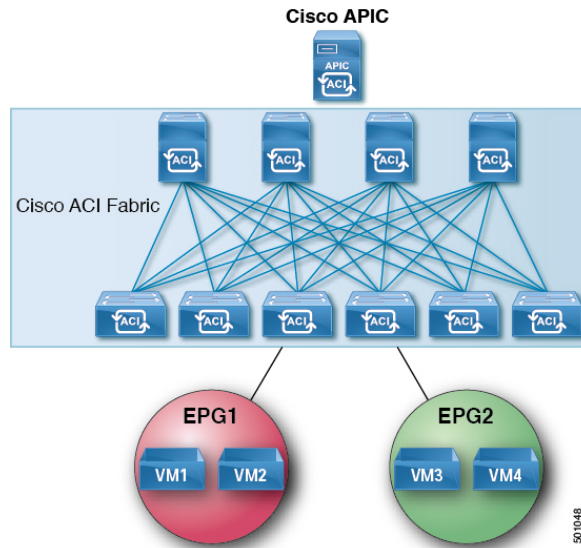
Figure 7: Proxy ARP and Cisco APIC



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 8: VM1 to VM2 Communication is Desired.**Table 3: ARP Table State**

Device	State
VM1	IP = * MAC = *
ACI fabric	IP = * MAC = *
VM2	IP = * MAC = *

2. VM1 sends an ARP request with a broadcast MAC address to VM2.

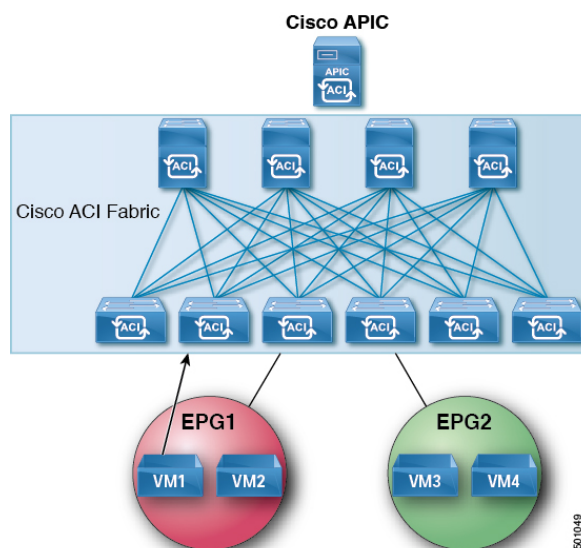
Figure 9: VM1 sends an ARP Request with a Broadcast MAC address to VM2

Table 4: ARP Table State

Device	State
VM1	IP = VM2 IP; MAC = ?
ACI fabric	IP = VM1 IP; MAC = VM1 MAC
VM2	IP = * MAC = *

3. The ACI fabric floods the proxy ARP request within the bridge domain (BD).

Figure 10: ACI Fabric Floods the Proxy ARP Request within the BD

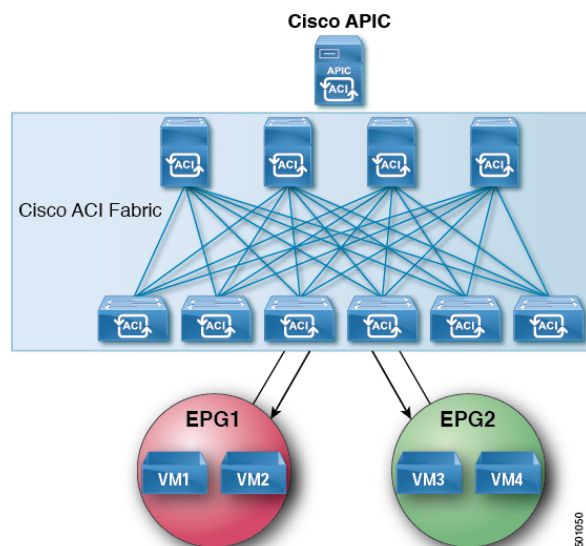
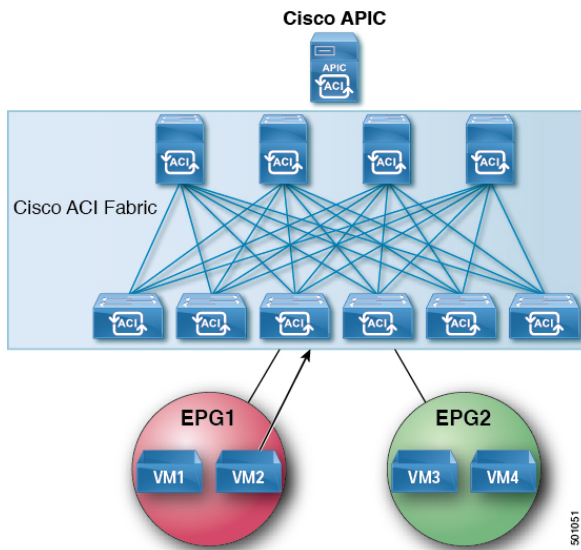


Table 5: ARP Table State

Device	State
VM1	IP = VM2 IP; MAC = ?
ACI fabric	IP = VM1 IP; MAC = VM1 MAC
VM2	IP = VM1 IP; MAC = BD MAC

4. VM2 sends an ARP response to the ACI fabric.

Figure 11: VM2 Sends an ARP Response to the ACI Fabric**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

5. VM2 is learned.

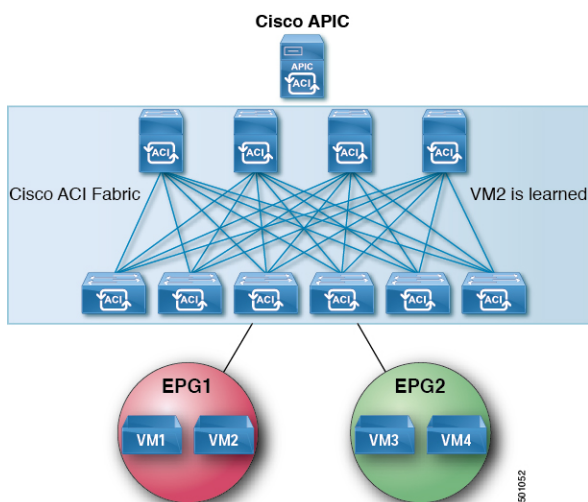
Figure 12: VM2 is Learned

Table 7: 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

6. VM1 sends an ARP request with a broadcast MAC address to VM2.

Figure 13: VM1 Sends an ARP Request with a Broadcast MAC Address to VM2

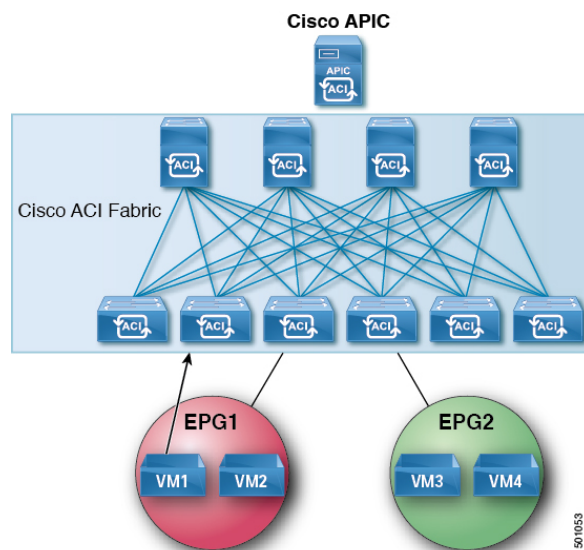


Table 8: ARP Table State

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

7. The ACI fabric sends a proxy ARP response to VM1.

Figure 14: ACI Fabric Sends a Proxy ARP Response to VM1

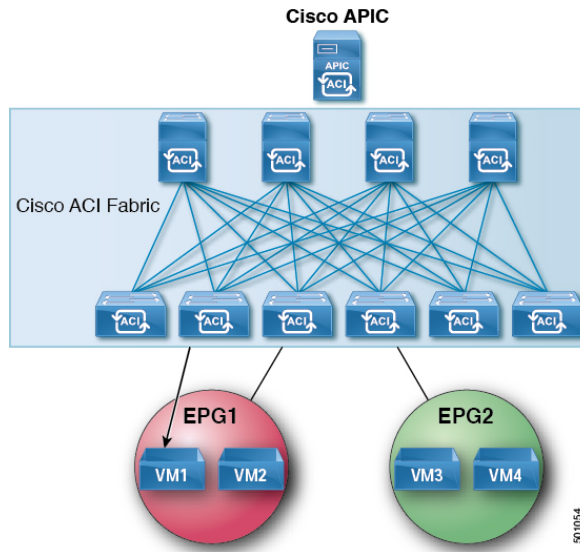


Table 9: ARP Table State

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 Cisco NX-OS Style CLI

Before you begin

- The appropriate tenant, VRF, bridge domain, application profile and EPG must be created.

- Intra-EPG isolation must be enabled on the EPG where proxy ARP has to be enabled.

Procedure

	Command or Action	Purpose
Step 1	configure Example: <code>apicl# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apicl(config)# tenant Tenant1</code>	Enters the tenant configuration mode.
Step 3	application <i>application-profile-name</i> Example: <code>apicl(config-tenant)# application Tenant1-App</code>	Creates an application profile and enters the application mode.
Step 4	epg <i>application-profile-EPG-name</i> Example: <code>apicl(config-tenant-app)# epg Tenant1-epg1</code>	Creates an EPG and enter the EPG mode.
Step 5	proxy-arp enable Example: <code>apicl(config-tenant-app-epg)# proxy-arp enable</code>	Enables proxy ARP. Note You can disable proxy-arp with the no proxy-arp command.
Step 6	exit Example: <code>apicl(config-tenant-app-epg)# exit</code>	Returns to application profile mode.
Step 7	exit Example: <code>apicl(config-tenant-app)# exit</code>	Returns to tenant configuration mode.
Step 8	exit Example: <code>apicl(config-tenant)# exit</code>	Returns to global configuration mode.

Examples

This example shows how to configure proxy ARP.

```

apicl# conf t
apicl(config)# tenant Tenant1
apicl(config-tenant)# application Tenant1-App

```

```

apic1(config-tenant-app)# epg Tenant1-epg1
apic1(config-tenant-app-epg)# proxy-arp enable
apic1(config-tenant-app-epg)#
apic1(config-tenant)#

```

Configuring Flood in Encapsulation

The configuration for Layer 2 external connectivity is similar to a static application EPG, where you map a VLAN on a node port to an EPG and map the EPG to a bridge-domain to provide/consume contracts.

SUMMARY STEPS

1. configure
2. tenant *tenant-name*
3. application *application-profile-name*
4. epg *application-profile-EPG-name*
5. flood-on-encapsulation enable
6. exit

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: apic1(config)# tenant Tenant1	Enters the tenant configuration mode.
Step 3	application <i>application-profile-name</i> Example: apic1(config)# application Tenant1-App	Creates an application profile and enters the application mode.
Step 4	epg <i>application-profile-EPG-name</i> Example: apic1(config)# epg Tenant1-epg1	Creates an EPG and enter the EPG mode.
Step 5	flood-on-encapsulation enable Example: apic1(config-tenant-app-epg)# flood-on-encapsulation enable	Enables flood-on-encapsulation.
Step 6	exit Example: apic1(config-tenant-app-epg)# exit	Returns to application profile mode.

Configuring 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.

Storm Control Guidelines

Configure traffic storm control levels according to the following guidelines and limitations:

- Typically, a fabric administrator configures storm control in fabric access policies on the following interfaces:
 - A regular trunk interface.
 - A direct port channel on a single leaf switch.
 - A virtual port channel (a port channel on two leaf switches).
- For port channels and virtual port channels, the storm control values (packets per second or percentage) apply to all individual members of the port channel. Do not configure storm control on interfaces that are members of a port channel.

**Note**

On switch hardware starting with the APIC 1.3(x) and switch 11.3(x) release, for port channel configurations, the traffic suppression on the aggregated port may be up to two times the configured value. The new hardware ports are internally subdivided into these two groups: slice-0 and slice-1. To check the slicing map, use the `vsh_lc` command `show platform internal hal l2 port gpd` and look for `slice 0` or `slice 1` under the `sl` column. If port-channel members fall on both slice-0 and slice-1, allowed storm control traffic may become twice the configured value because the formula is calculated based on each slice.

- When configuring by percentage of available bandwidth, a value of 100 means no traffic storm control and a value of 0.01 suppresses all traffic.
- Due to hardware limitations and the method by which packets of different sizes are counted, the level percentage is an approximation. Depending on the sizes of the frames that make up the incoming traffic, the actual enforced level might differ from the configured level by several percentage points. Packets-per-second (PPS) values are converted to percentage based on 256 bytes.
- Maximum burst is the maximum accumulation of rate that is allowed when no traffic passes. When traffic starts, all the traffic up to the accumulated rate is allowed in the first interval. In subsequent intervals, traffic is allowed only up to the configured rate. The maximum supported is 65535 KB. If the configured rate exceeds this value, it is capped at this value for both PPS and percentage.
- The maximum burst that can be accumulated is 512 MB.
- On an egress leaf switch in optimized multicast flooding (OMF) mode, traffic storm control will not be applied.
- On an egress leaf switch in non-OMF mode, traffic storm control will be applied.
- On a leaf switch for FEX, traffic storm control is not available on host-facing interfaces.
- Traffic storm control unicast/multicast differentiation is not supported on Cisco Nexus C93128TX, C9396PX, C9396TX, C93120TX, C9332PQ, C9372PX, C9372TX, C9372PX-E, or C9372TX-E switches.

Configuring a Traffic Storm Control Policy Using the NX-OS Style CLI

SUMMARY STEPS

1. Enter the following commands to create a PPS policy:
2. Enter the following commands to create a percent policy:
3. Configure storm control on physical ports, port channels, or virtual port channels:

DETAILED STEPS

	Command or Action	Purpose
Step 1	Enter the following commands to create a PPS policy: Example: <pre>(config)# template policy-group pg1 (config-pol-grp-if)# storm-control pps 10000 burst-rate 10000</pre>	
Step 2	Enter the following commands to create a percent policy: Example: <pre>(config)# template policy-group pg2 (config-pol-grp-if)# storm-control level 50 burst-rate 60</pre>	
Step 3	Configure storm control on physical ports, port channels, or virtual port channels: Example:	

Command or Action	Purpose
<pre>[no] storm-control [unicast multicast broadcast] level <percentage> [burst-rate <percentage>] [no] storm-control [unicast multicast broadcast] pps <packet-per-second> [burst-rate <packet-per-second>] sd-tb2-ifc1# configure terminal sd-tb2-ifc1(config)# leaf 102 sd-tb2-ifc1(config-leaf)# interface ethernet 1/19 sd-tb2-ifc1(config-leaf-if)# storm-control unicast level 35 burst-rate 45 sd-tb2-ifc1(config-leaf-if)# storm-control broadcast level 36 burst-rate 36 sd-tb2-ifc1(config-leaf-if)# storm-control broadcast level 37 burst-rate 38 sd-tb2-ifc1(config-leaf-if)# sd-tb2-ifc1# configure terminal sd-tb2-ifc1(config)# leaf 102 sd-tb2-ifc1(config-leaf)# interface ethernet 1/19 sd-tb2-ifc1(config-leaf-if)# storm-control broadcast pps 5000 burst-rate 6000 sd-tb2-ifc1(config-leaf-if)# storm-control unicast pps 7000 burst-rate 7000 sd-tb2-ifc1(config-leaf-if)# storm-control unicast pps 8000 burst-rate 10000 sd-tb2-ifc1(config-leaf-if)#</pre>	

Configuring 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 heartbeats (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:

- Fex ports are not supported for MACsec.
- Must-secure mode is not supported at Pod level.
- A MACsec policy with name 'default' is not supported.
- Auto key generation is only supported at the Pod level for fabric ports.
- MACsec is not supported on remote leaf switches.
- 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 which is running MACsec, **must-secure** mode requires changing the mode to should-secure in order for the node to join the Pod.
- Upgrade/Downgrade should only be initiated if the fabric links are in **should-secure** mode. Once upgrade/downgrade has completed, then the mode can be changed to **must-secure**. Upgrading/Downgrading in **must-secure** mode will result in nodes losing connectivity to the fabric. Recovering from connectivity loss requires that the fabric links of the nodes visible to the APIC be configured to **should-secure** mode. 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 PC/VPC interface, MACsec can be deployed via policy groups per PC/VPC interface. Port selectors are used to deploy the policies to a particular set of ports. Therefore, it is the user's responsibility to create the right port selector corresponding to the L3Out interfaces.
- It is recommended that MACsec policies be configured to **should-secure** mode before a configuration is exported.
- All the links on a spine are considered fabric links. However, if a spine link is used for IPN connectivity, then this link will be treated as an access link. This means that MACsec access policy needs to be used to deploy MACsec on these links.
- MACSEC Sessions may 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 keychain.

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 NX-OS Style CLI

Step 1 Configure MACsec Security Policy for access interfaces

Example:

```
apicl# configure
apicl(config)# template macsec access security-policy accmacsecpoll
apicl(config-macsec-param)# cipher-suite gcm-aes-128
apicl(config-macsec-param)# conf-offset offset-30
apicl(config-macsec-param)# description 'description for mac sec parameters'
apicl(config-macsec-param)# key-server-priority 1
apicl(config-macsec-param)# sak-expiry-time 110
apicl(config-macsec-param)# security-mode must-secure
apicl(config-macsec-param)# window-size 1
apicl(config-macsec-param)# exit
apicl(config)#
```

Step 2 Configure MACsec key chain for access interface:

PSK can be configured in 2 ways:

- Note**
- Inline with the **psk-string** command as illustrated in key 12ab below. The PSK is not secure because it is logged and exposed.
 - Entered separately in a new command **Enter PSK string** after the **psk-string** command as illustrated in key ab12. The PSK is secured because it is only echoed locally and is not logged.

Example:

```
apicl# configure
apicl(config)# template macsec access keychain acckeychainpoll
apicl(config-macsec-keychain)# description 'macsec key chain kcl'
```

```

apic1(config-macsec-keychain)# key 12ab
apic1(config-macsec-keychain-key)# life-time start 2017-09-19T12:03:15 end 2017-12-19T12:03:15
apic1(config-macsec-keychain-key)# psk-string 123456789a223456789a323456789abc
apic1(config-macsec-keychain-key)# exit
apic1(config-macsec-keychain)# key ab12
apic1(config-macsec-keychain-key)# life-time start now end infinite
apic1(config-macsec-keychain-key)# life-time start now end infinite
apic1(config-macsec-keychain-key)# psk-string
Enter PSK string: 123456789a223456789a323456789abc
apic1(config-macsec-keychain-key)# exit
apic1(config-macsec-keychain)# exit
apic1(config)#

```

Step 3 Configure MACsec interface policy for access interface:

Example:

```

apic1# configure
apic1(config)# template macsec access interface-policy accmacsecifpoll
apic1(config-macsec-if-policy)# inherit macsec security-policy accmacsecpoll keychain
acckeychainpoll
apic1(config-macsec-if-policy)# exit
apic1(config)#

```

Step 4 Associate MACsec interface policy to access interfaces on leaf (or spine):

Example:

```

apic1# configure
apic1(config)# template macsec access interface-policy accmacsecifpoll
apic1(config-macsec-if-policy)# inherit macsec security-policy accmacsecpoll keychain
acckeychainpoll
apic1(config-macsec-if-policy)# exit
apic1(config)#

```

Step 5 Configure MACsec Security Policy for fabric interfaces:

Example:

```

apic1# configure
apic1(config)# template macsec fabric security-policy fabmacsecpoll
apic1(config-macsec-param)# cipher-suite gcm-aes-xpn-128
apic1(config-macsec-param)# description 'description for mac sec parameters'
apic1(config-macsec-param)# window-size 1
apic1(config-macsec-param)# sak-expiry-time 100
apic1(config-macsec-param)# security-mode must-secure
apic1(config-macsec-param)# exit
apic1(config)#

```

Step 6 Configure MACsec key chain for fabric interface:

PSK can be configured in 2 ways:

- Note**
- Inline with the **psk-string** command as illustrated in key 12ab below. The PSK is not secure because it is logged and exposed.
 - Entered separately in a new command **Enter PSK string** after the **psk-string** command as illustrated in key ab12. The PSK is secured because it is only echoed locally and is not logged.

Example:

```

apic1# configure
apic1(config)# template macsec fabric security-policy fabmacsecpoll
apic1(config-macsec-param)# cipher-suite gcm-aes-xpn-128
apic1(config-macsec-param)# description 'description for mac sec parameters'

```

```
apicl(config-macsec-param)# window-size 1
apicl(config-macsec-param)# sak-expiry-time 100
apicl(config-macsec-param)# security-mode must-secure
apicl(config-macsec-param)# exit
apicl(config)# template macsec fabric keychain fabkeychainpol1
apicl(config-macsec-keychain)# description 'macsec key chain kc1'
apicl(config-macsec-keychain)# key 12ab
apicl(config-macsec-keychain-key)# psk-string 123456789a223456789a323456789abc
apicl(config-macsec-keychain-key)# life-time start 2016-09-19T12:03:15 end 2017-09-19T12:03:15
apicl(config-macsec-keychain-key)# exit
apicl(config-macsec-keychain)# key cd78
apicl(config-macsec-keychain-key)# psk-string
Enter PSK string: 123456789a223456789a323456789abc
apicl(config-macsec-keychain-key)# life-time start now end infinite
apicl(config-macsec-keychain-key)# exit
apicl(config-macsec-keychain)# exit
apicl(config)#
```

Step 7 Associate MACsec interface policy to fabric interfaces on leaf (or spine):

Example:

```
apicl# configure
apicl(config)# leaf 101
apicl(config-leaf)# fabric-interface ethernet 1/52-53
apicl(config-leaf-if)# inherit macsec interface-policy fabmacsecifpol2
apicl(config-leaf-if)# exit
apicl(config-leaf)#
```



CHAPTER 7

Configuring Layer 3 External Connectivity

- [About the Modes of Configuring Layer 3 External Connectivity, on page 163](#)
- [Configuring Layer 3 External Connectivity, on page 165](#)
- [Routed Connectivity to External Networks, on page 165](#)
- [Layer 3 Routed and Sub-Interface Port Channels, on page 178](#)
- [Layer 3 Out to Layer 3 Out Inter-VRF Leaking, on page 185](#)
- [About SVI External Encapsulation Scope, on page 189](#)
- [About SVI Auto State , on page 192](#)
- [Configuring an Interface and Static Route , on page 194](#)
- [OSPF Configuration, on page 197](#)
- [BGP Configuration, on page 205](#)
- [EIGRP Configuration, on page 223](#)
- [Configuring Route-Maps, on page 230](#)
- [Configuring Bi-Directional Route Forwarding \(BFD\), on page 242](#)
- [Configuring Layer 3 Multicast, on page 257](#)
- [Configuring External-L3 EPGs, on page 272](#)
- [Configuring Layer 3 External Connectivity Using the Named Mode, on page 275](#)
- [IPv6 Neighbor Discovery, on page 291](#)
- [Microsoft NLB, on page 296](#)
- [MLD Snooping, on page 299](#)
- [Configuring HSRP, on page 302](#)
- [Cisco ACI GOLF, on page 305](#)
- [Multipod_Fabric, on page 322](#)
- [Remote Leaf Switches, on page 328](#)
- [Transit Routing, on page 336](#)

About the Modes of Configuring Layer 3 External Connectivity

Because APIC supports multiple user interfaces (UIs) for configuration, the potential exists for unintended interactions when you create a configuration with one UI and later modify the configuration with another UI. This section describes considerations for configuring Layer 3 external connectivity with the APIC NX-OS style CLI, when you may also be using other APIC user interfaces.

When you configure Layer 3 external connectivity with the APIC NX-OS style CLI, you have the choice of two modes:

- Implicit mode, a simpler mode, is not compatible with the APIC GUI or the REST API.
- Named (or Explicit) mode is compatible with the APIC GUI and the REST API.

In either case, the configuration should be considered read-only in the incompatible UI.

How the Modes Differ

In both modes, the configuration settings are defined within an internal container object, the "L3 Outside" (or "L3Out"), which is an instance of the **l3extOut** class in the API. The main difference between the two modes is in the naming of this container object instance:

- Implicit mode—the naming of the container is implicit and does not appear in the CLI commands. The CLI creates and maintains these objects internally.
- Named mode—the naming is provided by the user. CLI commands in the Named Mode have an additional **l3Out** field. To configure the named L3Out correctly and avoid faults, the user is expected to understand the API object model for external Layer 3 configuration.



Note

Except for the procedures in the *Configuring Layer 3 External Connectivity Using the Named Mode* section, this guide describes Implicit mode procedures.

Guidelines and Restrictions

- In the same APIC instance, both modes can be used together for configuring Layer 3 external connectivity with the following restriction: The Layer 3 external connectivity configuration for a given combination of tenant, VRF, and leaf can be done only through one mode.
- For a given tenant VRF, the policy domain where the External-l3 EPG can be placed can be in either the Named mode or in the Implicit mode. The recommended configuration method is to use only one mode for a given tenant VRF combination across all the nodes where the given tenant VRF is deployed for Layer 3 external connectivity. The modes can be different across different tenants or different VRFs and no restrictions apply.
- In some cases, an incoming configuration to a Cisco APIC cluster will be validated against inconsistencies, where the validations involve externally-visible configurations (northbound traffic through the L3Outs). An Invalid Configuration error message will appear for those situations where the configuration is invalid.
- The external Layer 3 features are supported in both configuration modes, with the following exception:
 - Route-peering and Route Health Injection (RHI) with a L4-L7 Service Appliance is supported only in the Named mode. The Named mode should be used across all border leaf switches for the tenant VRF where route-peering is involved.
- Layer 3 external network objects (l3extOut) created using the Implicit mode CLI procedures are identified by names starting with “_ui_” and are marked as read-only in the GUI. The CLI partitions these external-l3 networks by function, such as interfaces, protocols, route-map, and EPG. Configuration modifications performed through the REST API can break this structure, preventing further modification through the CLI.

For the steps to remove such objects, see *Troubleshooting Unwanted _ui_ Objects* in the *APIC Troubleshooting Guide*.

Configuring Layer 3 External Connectivity

Configuration of layer 3 (L3) routing connectivity to an external network consists of the following components:

- **Interface**—Interface configuration for layer 3 ports, sub-interfaces, external SVI that are used to connect to external routers.
- **Routing Protocol Configuration**—CLI supports static route, BGP, OSPF, EIGRP protocol configuration.
- **Route-map control**—A route map is used to match prefixes/BD public subnets and apply route-control policies. Once created, it can be associated with routing protocols in a direction, such as “in” (BGP or OSPF), “out”(BGP, OSPF, EIGRP).

Configurations pertaining to interface, routing protocols, and route-maps are maintained per leaf switch under the config-leaf configuration mode.

- **External-L3 EPG**—A list of external subnets on a tenant VRF that are classified as one endpoint group for applying contract and QoS policies. External-L3 EPGs (also called prefix EPGs) can have contracts with other external-L3 EPGs and application EPGs. External-L3 EPG configuration is maintained under tenant configuration. The external-L3 EPGs can be deployed on a subset of nodes where the VRF is configured.

The steps for configuring layer 3 external connectivity can be summarized as follows:

1. Create a VRF under a tenant.
2. Configure and deploy the VRF on the border leaf switch.
3. Configure layer 3 interfaces on the border leaf Interfaces.
4. Configure route-maps on the leaf switch.
5. Configure routing protocols (BGP, OSPF, EIGRP) under leaf and leaf-interface.
6. Create and configure an external-L3 EPG under a tenant.
7. Deploy the external-L3 EPG on the border leaf switch.

Routed Connectivity to External Networks

About Routed Connectivity to Outside Networks

A Layer 3 outside network configuration (L3Out) defines how traffic is forwarded outside of the fabric. Layer 3 is used to discover the addresses of other nodes, select routes, select quality of service, and forward the traffic that is entering, exiting, and transiting the fabric.

**Note**

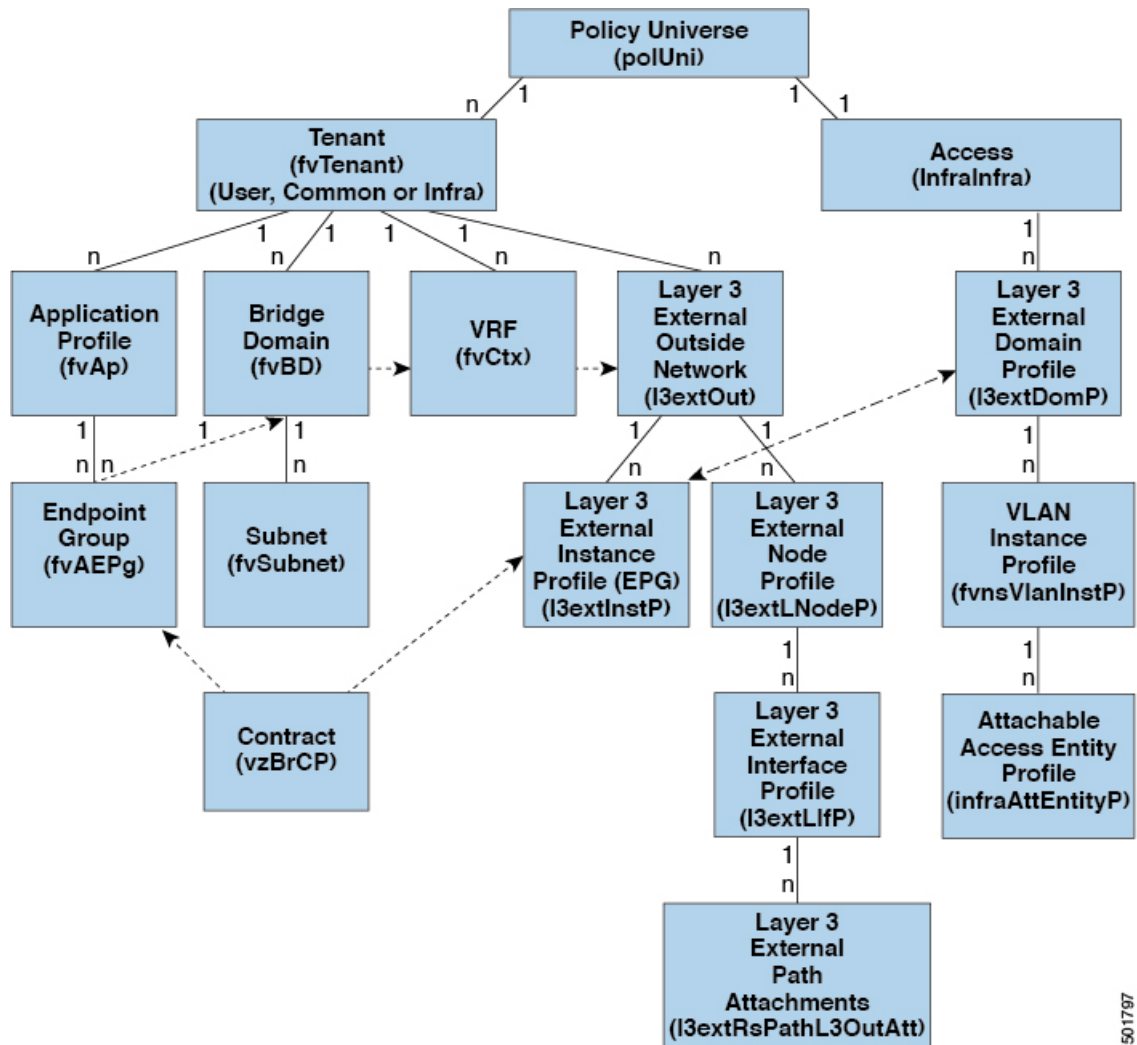
For guidelines and cautions for configuring and maintaining Layer 3 outside connections, see [Guidelines for Routed Connectivity to Outside Networks](#), on page 168.

For information about the types of L3Outs, see [External Layer 3 Outside Connection Types](#), on page 170.

Layer 3 Out for Routed Connectivity to External Networks

Routed connectivity to external networks is enabled by associating a fabric access (*infraInfra*) external routed domain (*l3extDomP*) with a tenant Layer 3 external instance profile (*l3extInstP* or external EPG) of a Layer 3 external outside network (*l3extOut*), in the hierarchy in the following diagram:

Figure 15: Policy Model for Layer 3 External Connections



A Layer 3 external outside network (*l3extOut* object) includes the routing protocol options (BGP, OSPF, or EIGRP or supported combinations) and the switch-specific and interface-specific configurations. While the *l3extOut* contains the routing protocol (for example, OSPF with its related Virtual Routing and Forwarding (VRF) and area ID), the Layer 3 external interface profile contains the necessary OSPF interface details. Both are needed to enable OSPF.

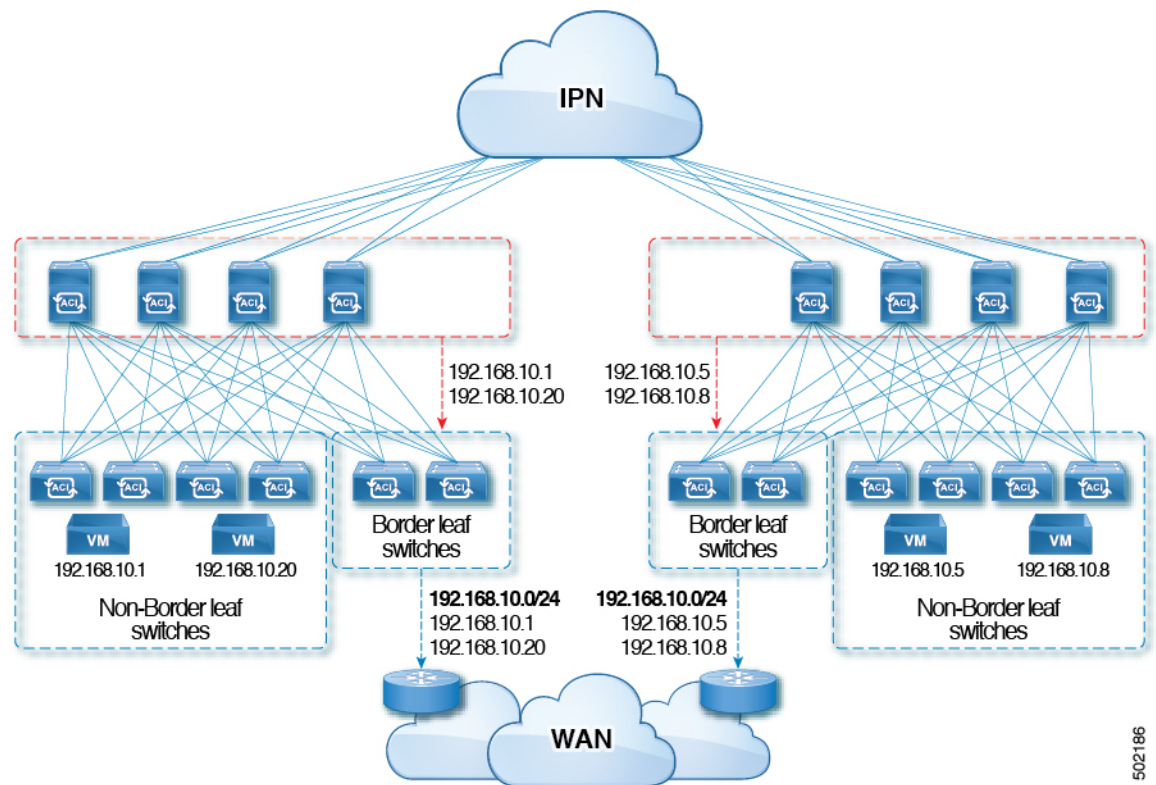
The *l3extInstP* EPG exposes the external network to tenant EPGs through a contract. For example, a tenant EPG that contains a group of web servers could communicate through a contract with the *l3extInstP* EPG according to the network configuration contained in the *l3extOut*. The outside network configuration can easily be reused for multiple nodes by associating the nodes with the L3 external node profile. Multiple nodes that use the same profile can be configured for fail-over or load balancing. Also, a node can be added to

multiple l3extOuts resulting in VRFs that are associated with the l3extOuts also being deployed on that node. For scalability information, refer to the current *Verified Scalability Guide for Cisco ACI*.

Advertise Host Routes

Enabling Advertise Host Routes on the BD, individual host-routes (/32 and /128 prefixes) are advertised from the Border-Leaf switches (BL). The BD must be associated to the L3out or an explicit prefix list matching the host routes. The host routes must be configured to advertise host routes out of the fabric.

Border-Leaf switches along with the subnet advertise the individual end-point (EP) prefixes. The route information is advertised only if the host is connected to the local POD. If the EP is moved away from the local POD or once the EP is removed from EP database (even if the EP is attached to a remote leaf), the route advertisement is then withdrawn.



Advertise Host Route configuration guidelines and limitations are:

- Enabling PIMv4 (Protocol-Independent Multicast, version 4) and Advertise Host routes on a BD is not supported.
- Host route advertisement supports both BD to L3out Association and the explicit route map configurations. We recommend using explicit route map configuration which allows you greater control in selecting individual or a range of host routes to configure.
- EPs/Host routes in SITE-1 will not be advertised out through Border Leafs in other SITES.
- When EPs is aged out or removed from the database, Host routes are withdrawn from the Border Leaf.
- When EP is moved across SITES or PODs, Host routes should be withdrawn from first SITE/POD and advertised in new POD/SITE.

- EPs learned on a specific BD, under any of the BD subnets are advertised from the L3out on the border leaf in the same POD.
- EPs are advertised out as Host Routes only in the local POD through the Border Leaf.
- Host routes are not advertised out from one POD to another POD.
- In the case of Remote Leaf, if EPs are locally learned in the Remote Leaf, they are then advertised only through a L3out deployed in Remote Leaf switches in same POD.
- EPs/Host routes in a Remote Leaf are not advertised out through Border Leaf switches in main POD or another POD.
- EPs/Host routes in the main POD are not advertised through L3out in Remote Leaf switches of same POD or another POD.
- The BD subnet must have the **Advertise Externally** option enabled.
- The BD must be associated to an L3out or the L3out must have explicit route-map configured matching BD subnets.
- There must be a contract between the EPG in the specified BD and the External EPG for the L3out.

**Note**

If there is no contract between the BD/EPG and the External EPG the BD subnet and host routes will not be installed on the border leaf.

- Advertise Host Route is supported for shared services. For example: epg1/BD1 deployed is in VRF-1 and L3out in another VRF-2. By providing shared contract between EPG and L3out host routes are pulled from one VRF-1 to another VRF-2.
- When Advertise Host Route is enabled on BD custom tag cannot be set on BD Subnet using route-map.
- When Advertise Host Route is enabled on a BD and the BD is associated with an L3Out, BD subnet is marked public. If there's a rogue EP present under the BD, that EP is advertised out on L3Out.

Guidelines for Routed Connectivity to Outside Networks

Use the following guidelines when creating and maintaining Layer 3 outside connections.

Topic	Caution or Guideline
Ingress-based policy enforcement	Starting with Cisco APIC release 1.2(1), ingress-based policy enforcement enables defining policy enforcement for Layer 3 Outside (L3Out) traffic for both egress and ingress directions. The default is ingress. During an upgrade to release 1.2(1) or higher, existing L3Out configurations are set to egress so that the behavior is consistent with the existing configuration. You do not need any special upgrade sequence. After the upgrade, you change the global property value to ingress. When it has been changed, the system reprograms the rules and prefix entries. Rules are removed from the egress leaf and installed on the ingress leaf, if not already present. If not already configured, an <code>Actrl</code> prefix entry is installed on the ingress leaf. Direct server return (DSR), and attribute EPGs require ingress based policy enforcement. <code>vzAny</code> and <code>taboo</code> contracts ignore ingress based policy enforcement. Transit rules are applied at ingress.
Bridge Domains with L3Outs	A bridge domain in a tenant can contain a public subnet that is advertised through an <code>l3extOut</code> provisioned in the common tenant.
Bridge domain route advertisement For OSPF and EIGRP	When both OSPF and EIGRP are enabled on the same VRF on a node and if the bridge domain subnets are advertised out of one of the L3Outs, it will also get advertised out of the protocol enabled on the other L3Out. For OSPF and EIGRP, the bridge domain route advertisement is per VRF and not per L3Out. The same behavior is expected when multiple OSPF L3Outs (for multiple areas) are enabled on the same VRF and node. In this case, the bridge domain route will be advertised out of all the areas, if it is enabled on one of them.
BGP Maximum Prefix Limit	Starting with Cisco APIC release 1.2(1x), tenant policies for BGP <code>l3extOut</code> connections can be configured with a maximum prefix limit, that enables monitoring and restricting the number of route prefixes received from a peer. Once the maximum prefix limit has been exceeded, a log entry is recorded, and further prefixes are rejected. The connection can be restarted if the count drops below the threshold in a fixed interval, or the connection is shut down. Only one option can be used 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, before the APIC raises a fault.

Topic	Caution or Guideline
MTU	Cisco ACI does not support IP fragmentation. Therefore, when you configure Layer 3 Outside (L3Out) connections to external routers, or multipod connections through an Inter-Pod Network (IPN), it is critical 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. Cisco highly recommends that you test the MTU with CLI-based commands. For example, on the Cisco NX-OS CLI, use a command such as <code>ping 1.1.1.1 df-bit packet-size 9000 source-interface ethernet 1/1</code>.
Layer 4 to Layer 7	When you are using a multinode service graph, you must have the two EPGs in separate VRF instances. For these functions, the system must do a Layer 3 lookup, so the EPGs must be in separate VRFs. This limitation follows legacy service insertion, based on Layer 2 and Layer 3 lookups.
QoS for L3Outs	To configure QoS policies for an L3Out and enable the policies to be enforced on the BL switch where the L3Out is located, use the following guidelines: • The VRF Policy Control Enforcement Direction must be set to Egress. • The VRF Policy Control Enforcement Preference must be set to Enabled. • When configuring the contract that controls communication between the EPGs using the L3Out, include the QoS class or Target DSCP in the contract or subject of the contract.

External Layer 3 Outside Connection Types

ACI supports the following External Layer 3 Outside connection options:

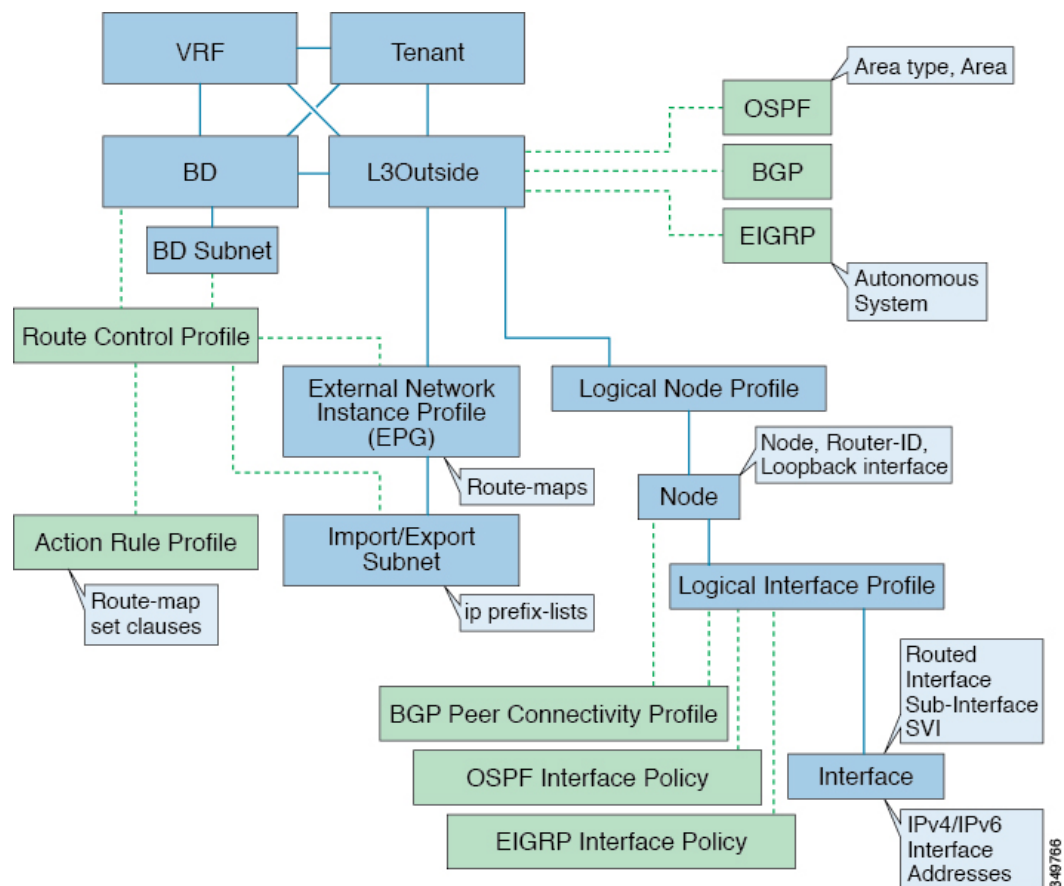
- Static Routing (supported for IPv4 and IPv6)
- OSPFv2 for normal and NSSA areas (IPv4)
- OSPFv3 for normal and NSSA areas (IPv6)
- iBGP (IPv4 and IPv6)

- eBGP (IPv4 and IPv6)
- EIGRP (IPv4 and IPv6)

The External Layer 3 Outside connections are supported on the following interfaces:

- Layer 3 Routed Interface
- Sub-interface with 802.1Q tagging - With sub-interface, the same physical interface can be used to provide a Layer 2 outside connection for multiple private networks.
- Switched Virtual Interface (SVI) - With an SVI interface, the same physical interface that supports Layer 2 and Layer 3 and the same physical interface can be used for a Layer 2 outside connection and a Layer 3 outside connection.

Figure 16: ACI Layer 3 Managed Objects



The managed objects that are used for the L3Outside connections are:

- External Layer 3 Outside (L3ext): Routing protocol options (OSPF area type, area, EIGRP AS, BGP), private network, External Physical domain.
- Logical Node Profile: Profile where one or more nodes are defined for the External Layer 3 Outside connections. The router-IDs and the loopback interface configuration is defined in the profile.



Note Use the same router-ID for the same node across multiple External Layer 3 Outside connections.



Note Within a single L3Out, a node can only be part of one Logical Node Profile. Configuring the node to be a part of multiple Logical Node Profiles in a single L3Out might result in unpredictable behavior, such as a loopback address being pushed from one Logical Node Profile but not from the other. Use additional path bindings under the existing Logical Interface Profiles or create a new Logical Interface Profile under the existing Logical Node Profile instead.

- Logical Interface Profile: IP interface configuration for IPv4 and IPv6 interfaces. It is supported on the Route Interfaces, Routed Sub-Interfaces, and SVIs. The SVIs can be configured on physical ports, port-channels or VPCs.
- OSPF Interface Policy: Includes details such as OSPF Network Type and priority.
- EIGRP Interface Policy: Includes details such as Timers and split horizon.
- BGP Peer Connectivity Profile: The profile where most BGP peer settings, remote-as, local-as, and BGP peer connection options are configured. The BGP peer connectivity profile can be associated with the logical interface profile or the loopback interface under the node profile. This determines the update-source configuration for the BGP peering session.
- External Network Instance Profile (EPG) (l3extInstP): The external EPG is also referred to as the prefix based EPG or InstP. The import and export route control policies, security import policies, and contract associations are defined in this profile. Multiple external EPGs can be configured under a single L3Out. Multiple external EPGs may be used when a different route or a security policy is defined on a single External Layer 3 Outside connections. An external EPG or multiple external EPGs combine into a route-map. The import/export subnets defined under the external EPG associate to the IP prefix-list match clauses in the route-map. The external EPG is also where the import security subnets and contracts are associated. This is used to permit or drop traffic for this L3out.
- Action Rules Profile: The action rules profile is used to define the route-map set clauses for the L3Out. The supported set clauses are the BGP communities (standard and extended), Tags, Preference, Metric, and Metric type.
- Route Control Profile: The route-control profile is used to reference the action rules profile(s). This can be an ordered list of action rules profiles. The Route Control Profile can be referenced by a tenant BD, BD subnet, external EPG, or external EPG subnet.

There are additional protocol settings for BGP, OSPF, and EIGRP L3Outs. These settings are configured per tenant in the ACI Protocol Policies section in the GUI.



Note When configuring policy enforcement between external EPGs (transit routing case), you must configure the second external EPG (InstP) with the default prefix 0/0 for export route control, aggregate export, and external security. In addition, the preferred group must be excluded, and you must use an any contract (or desired contract) between the transit InstPs.

Configuring a Layer 3 Outside for Tenant Networks Using the NX-OS Style CLI

These steps describe how to configure a Layer 3 outside network for tenant networks. This example shows how to deploy a node and L3 port for tenant VRF external L3 connectivity using the NX-OS CLI.

This example is broken into steps for clarity. For a merged example, see [NX-OS Style CLI Example: L3Out, on page 176](#).

Before you begin

- Configure the node, port, functional profile, AEP, and Layer 3 domain.
- Configure a VLAN domain using the **vlan-domain** *domain* and **vlan** *vlan-range* commands.
- Configure a BGP route reflector policy to propagate the routed within the fabric.

For an example using the commands for these prerequisites, see [NX-OS Style CLI Example: L3Out Prerequisites, on page 176](#).

Step 1

Configure the tenant and VRF.

This example configures tenant `t1` with VRF `v1`. They are not yet deployed.

Example:

```
apic1# configure
apic1(config)# tenant t1
apic1(config-tenant)# vrf context v1
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# exit
apic1(config)#
```

Step 2

Configure the node and interface for the L3Out.

This example configures VRF `v1` on node 103 (the border leaf switch), which is named `nodep1`, with router ID `11.11.11.103`. It also configures interface `eth1/3` as a routed interface (Layer 3 port), with IP address `12.12.12.3/24` and Layer 3 domain `dom1`.

Example:

```
apic1(config)# leaf 103
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# router-id 11.11.11.103
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# interface ethernet 1/3
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member tenant t1 vrf v1
apic1(config-leaf-if)# ip address 12.12.12.3/24
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

Step 3

Configure the routing protocol.

This example configures BGP as the primary routing protocol, with a BGP peer address, `15.15.15.2` and ASN 100.

Example:

```
apic1(config)# leaf 103
apic1(config-leaf)# router bgp 100
```

```

apicl(config-leaf-bgp)# vrf member tenant t1 vrf v1
apicl(config-leaf-bgp-vrf)# neighbor 15.15.15.2
apicl(config-leaf-bgp-vrf-neighbor)# exit
apicl(config-leaf-bgp-vrf)# exit
apicl(config-leaf-bgp)# exit
apicl(config-leaf)# exit

```

Step 4 Optional. Configure a connectivity routing protocol.

This example configures OSPF as the communication protocol, with regular area ID 0.0.0.0, with loopback address 30.30.30.0.

Example:

```

apicl(config)# leaf 103
apicl(config-leaf)# router ospf default
apicl(config-leaf-ospf)# vrf member tenant t1 vrf v1
apicl(config-leaf-ospf-vrf)# area 0.0.0.0 loopback 30.30.30.0
apicl(config-leaf-ospf-vrf)# exit
apicl(config-leaf-ospf)# exit
apicl(config-leaf)# exit

```

Step 5 Configure the external EPG on node 103.

In this example, the network 20.20.20.0/24 is configured as the external network `extnwl`.

Example:

```

apicl(config)# tenant t1
apicl(config-tenant)# external-l3 epg extnwl
apicl(config-tenant-l3ext-epg)# vrf member v1
apicl(config-tenant-l3ext-epg)# match ip 20.20.20.0/24
apicl(config-tenant-l3ext-epg)# exit
apicl(config-tenant)# exit
apicl(config)# leaf 103
apicl(config-leaf)# vrf context tenant t1 vrf v1
apicl(config-leaf-vrf)# external-l3 epg extnwl
apicl(config-leaf-vrf)# exit

```

Step 6 Optional. Configure Advertise Host Routing.

Example:

```

apicl# configure
apicl(config)# tenant <Name>
apicl(config-tenant)# bridge-domain <Name>
apicl(config-tenant-bd)# advertise-host-routes
apicl(config-tenant-bd)# end

```

Step 7 Optional. Configure a route map.

This example configures a route map `rp1` 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:

```

apicl(config-leaf)# template route group match-rule1 tenant t1
apicl(config-route-group)# ip prefix permit 200.3.2.0/24
apicl(config-route-group)# exit
apicl(config-leaf)# vrf context tenant t1 vrf v1
apicl(config-leaf-vrf)# route-map rp1
apicl(config-leaf-vrf-route-map)# match route group match-rule1 order 0
apicl(config-leaf-vrf-route-map-match)# exit
apicl(config-leaf-vrf-route-map)# exit

```

```
apicl(config-leaf-vrf)# exit
apicl(config)# leaf 103
apicl(config-leaf)# router bgp 100
apicl(config-leaf-bgp)# vrf member tenant t1 vrf v1
apicl(config-leaf-bgp-vrf)# neighbor 15.15.15.2
apicl(config-leaf-bgp-vrf-neighbor)# route-map rp1 in
apicl(config-leaf-bgp-vrf-neighbor)# exit
apicl(config-leaf-bgp-vrf)# exit
apicl(config-leaf-bgp)# exit
apicl(config-leaf)# exit
```

Step 8 Add a bridge domain.

Example:

```
apicl(config)# tenant t1
apicl(config-tenant)# bridge-domain bd1
apicl(config-tenant-bd)# vrf member v1
apicl(config-tenant-bd)# exit
apicl(config-tenant)# interface bridge-domain bd1
apicl(config-tenant-interface)# ip address 44.44.44.1/24 scope public
apicl(config-tenant-interface)# exit
apicl(config-tenant)# exit
apicl(config)# leaf 101
apicl(config-leaf)# vrf context tenant t1 vrf v1
apicl(config-leaf-vrf)# route-map rp1
apicl(config-leaf-vrf-route-map)# match bridge-domain bd1 tenant t1
apicl(config-leaf-vrf-route-map-match)# exit
apicl(config-leaf-vrf-route-map)# exit
apicl(config-leaf-vrf)# exit
apicl(config-leaf)# exit
```

Step 9 Create an application EPG on node 101.

Example:

```
apicl(config)# tenant t1
apicl(config-tenant)# application appl
apicl(config-tenant-app)# epg epg1
apicl(config-tenant-app-epg)# bridge-domain member bd1
apicl(config-tenant-app-epg)# exit
apicl(config-tenant-app)# exit
apicl(config-tenant)# exit
apicl(config)# leaf 101
apicl(config-leaf)# interface ethernet 1/3
apicl(config-leaf-if)# vlan-domain member dom1
apicl(config-leaf-if)# switchport trunk allowed vlan 2011 tenant t1 application appl epg epg1
apicl(config-leaf-if)# exit
apicl(config-leaf)# exit
apicl(config)#
```

Step 10 Create filters (access-lists) and contracts.

Example:

```
apicl(config)# tenant t1
apicl(config-tenant)# access-list http-filter
apicl(config-tenant-acl)# match ip
apicl(config-tenant-acl)# match tcp dest 80
apicl(config-tenant-acl)# exit
apicl(config-tenant)# contract httpCtrct
apicl(config-tenant-contract)# scope vrf
apicl(config-tenant-contract)# subject subj1
apicl(config-tenant-contract-subj)# access-group http-filter both
apicl(config-tenant-contract-subj)# exit
apicl(config-tenant-contract)# exit
```

Step 11 Configure contracts and associate them with EPGs.

Example:

```
apic1(config-tenant)# external-l3 epg extnw1
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# contract provider httpCtrct
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# application appl
apic1(config-tenant-app)# epg epg1
apic1(config-tenant-app-epg)# contract consumer httpCtrct
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)# exit
apic1(config-tenant)# exit
apic1(config)#
```

NX-OS Style CLI Example: L3Out Prerequisites

Before you can configure an L3Out, perform the following steps:

1. Configure a VLAN domain:

```
apic1# configure
apic1(config)# vlan-domain dom1
apic1(config-vlan)# vlan 1024-2048
apic1(config-vlan)# exit
```

2. Configure BGP route reflectors:

```
apic1(config)# bgp-fabric
apic1(config-bgp-fabric)# asn 100
apic1(config-bgp-fabric)# route-reflector spine 104,105
```

NX-OS Style CLI Example: L3Out

The following example provides a merged version of the steps to configure an L3Out using the NX-OS style CLI. Configure the following prerequisites before configuring the L3Out.

```
apic1# configure
apic1(config)# tenant t1
apic1(config-tenant)# vrf context v1
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 103
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# router-id 11.11.11.103
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# interface ethernet 1/3
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member tenant t1 vrf v1
apic1(config-leaf-if)# ip address 12.12.12.3/24
apic1(config-leaf-if)# exit
apic1(config-leaf)# router bgp 100
apic1(config-leaf-bgp)# vrf member tenant t1 vrf v1
apic1(config-leaf-bgp-vrf)# neighbor 15.15.15.2
apic1(config-leaf-bgp-vrf-neighbor)# exit
apic1(config-leaf-bgp-vrf)# exit
```

```
apic1(config-leaf-bgp)# exit
apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant t1 vrf v1
apic1(config-leaf-ospf-vrf)# area 0.0.0.0 loopback 30.30.30.0
apic1(config-leaf-ospf-vrf)# exit
apic1(config-leaf-ospf)# exit
apic1(config-leaf)# exit
apic1(config)# tenant t1
apic1(config-tenant)# external-l3 epg extnw1
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# match ip 20.20.20.0/24
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 103
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# external-l3 epg extnw1
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# template route group match-rule1 tenant t1
apic1(config-route-group)# ip prefix permit 200.3.2.0/24
apic1(config-route-group)# exit
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# route-map rp1
apic1(config-leaf-vrf-route-map)# match route group match-rule1 order 0
apic1(config-leaf-vrf-route-map-match)# exit
apic1(config-leaf-vrf-route-map)# exit
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# router bgp 100
apic1(config-leaf-bgp)# vrf member tenant t1 vrf v1
apic1(config-leaf-bgp-vrf)# neighbor 15.15.15.2
apic1(config-leaf-bgp-vrf-neighbor)# route-map rp1 in
apic1(config-leaf-bgp-vrf-neighbor)# exit
apic1(config-leaf-bgp-vrf)# exit
apic1(config-leaf-bgp)# exit
apic1(config-leaf)# exit
apic1(config)# tenant t1
apic1(config-tenant)# bridge-domain bd1
apic1(config-tenant-bd)# vrf member v1
apic1(config-tenant-bd)# exit
apic1(config-tenant)# interface bridge-domain bd1
apic1(config-tenant-interface)# ip address 44.44.44.1/24 scope public
apic1(config-tenant-interface)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# route-map map1
apic1(config-leaf-vrf-route-map)# match bridge-domain bd1 tenant t1
apic1(config-leaf-vrf-route-map-match)# exit
apic1(config-leaf-vrf-route-map)# exit
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# exit
apic1(config)# tenant t1
apic1(config-tenant)# application appl
apic1(config-tenant-app)# epg epg1
apic1(config-tenant-app-epg)# bridge-domain member bd1
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/3
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk allowed vlan 2011 tenant t1 application appl epg
epg1
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

```

apic1(config)# tenant t1
apic1(config-tenant)# access-list http-filter
apic1(config-tenant-acl)# match ip
apic1(config-tenant-acl)# match tcp dest 80
apic1(config-tenant-acl)# exit
apic1(config-tenant)# contract httpCtrct
apic1(config-tenant-contract)# scope vrf
apic1(config-tenant-contract)# subject subj1
apic1(config-tenant-contract-subj)# access-group http-filter both
apic1(config-tenant-contract-subj)# exit
apic1(config-tenant-contract)# exit
apic1(config-tenant)# external-l3 epg extnw1
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# contract provider httpCtrct
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# application appl
apic1(config-tenant-app)# epg epg1
apic1(config-tenant-app-epg)# contract consumer httpCtrct
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)# exit
apic1(config-tenant)# exit
apic1(config)#

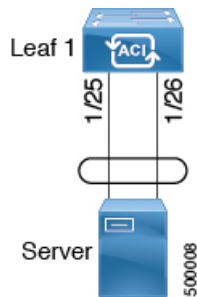
```

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 17: 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 a Layer 3 Routed Port-Channel Using the NX-OS CLI

This procedure configures a Layer 3 routed port channel.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **interface port-channel** *channel-name*
4. **no switchport**
5. **vrf member** *vrf-name* **tenant** *tenant-name*
6. **vlan-domain member** *vlan-domain-name*
7. **ip address** *ip-address* / *subnet-mask*
8. **ipv6 address** *sub-bits/prefix-length* **preferred**
9. **ipv6 link-local** *ipv6-link-local-address*
10. **mac-address** *mac-address*
11. **mtu** *mtu-value*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf switch or leaf switches to be configured. The <i>node-id</i> can be a single node ID or a range of IDs, in the form <i>node-id1-node-id2</i> , to which the configuration will be applied.
Step 3	interface port-channel <i>channel-name</i> Example: <code>apic1(config-leaf)# interface port-channel po1</code>	Enters the interface configuration mode for the specified port channel.
Step 4	no switchport Example: <code>apic1(config-leaf-if)# no switchport</code>	Makes the interface Layer 3 capable.
Step 5	vrf member <i>vrf-name</i> tenant <i>tenant-name</i> Example: <code>apic1(config-leaf-if)# vrf member v1 tenant t1</code>	Associates this port channel to this virtual routing and forwarding (VRF) instance and L3 outside policy, where: <i>vrf-name</i> is the VRF name. The name can be any case-sensitive, alphanumeric string up to 32 characters. <i>tenant-name</i> is the tenant name. The name can be any case-sensitive, alphanumeric string up to 32 characters.
Step 6	vlan-domain member <i>vlan-domain-name</i> Example: <code>apic1(config-leaf-if)# vlan-domain member dom1</code>	Associates the port channel template with the previously configured VLAN domain.

	Command or Action	Purpose
Step 7	ip address <i>ip-address / subnet-mask</i> Example: <code>apic1(config-leaf-if)# ip address 10.1.1.1/24</code>	Sets the IP address and subnet mask for the specified interface.
Step 8	ipv6 address <i>sub-bits/prefix-length preferred</i> Example: <code>apic1(config-leaf-if)# ipv6 address 2001::1/64 preferred</code>	Configures an IPv6 address based on an IPv6 general prefix and enables IPv6 processing on an interface, where: • <i>sub-bits</i> is the subprefix bits and host bits of the address to be concatenated with the prefixes provided by the general prefix specified with the prefix-name argument. The sub-bits argument must be in the form documented in RFC 2373 where the address is specified in hexadecimal using 16-bit values between colons. • <i>prefix-length</i> is the length of the IPv6 prefix. A decimal value that indicates how many of the high-order contiguous bits of the address comprise the prefix (the network portion of the address). A slash mark must precede the decimal value.
Step 9	ipv6 link-local <i>ipv6-link-local-address</i> Example: <code>apic1(config-leaf-if)# ipv6 link-local fe80::1</code>	Configures an IPv6 link-local address for an interface.
Step 10	mac-address <i>mac-address</i> Example: <code>apic1(config-leaf-if)# mac-address 00:44:55:66:55::01</code>	Manually sets the interface MAC address.
Step 11	mtu <i>mtu-value</i> Example: <code>apic1(config-leaf-if)# mtu 1500</code>	Sets the MTU for this class of service.

Example

This example shows how to configure a basic Layer 3 port channel.

```

apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface port-channel po1
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member vl tenant t1
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# ip address 10.1.1.1/24
apic1(config-leaf-if)# ipv6 address 2001::1/64 preferred
apic1(config-leaf-if)# ipv6 link-local fe80::1
apic1(config-leaf-if)# mac-address 00:44:55:66:55::01
apic1(config-leaf-if)# mtu 1500

```


Configuring a Layer 3 Sub-Interface Port-Channel Using the NX-OS CLI

This procedure configures a Layer 3 sub-interface port channel.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **vrf member** *vrf-name* **tenant** *tenant-name*
4. **vlan-domain member** *vlan-domain-name*
5. **ip address** *ip-address* / *subnet-mask*
6. **ipv6 address** *sub-bits/prefix-length* **preferred**
7. **ipv6 link-local** *ipv6-link-local-address*
8. **mac-address** *mac-address*
9. **mtu** *mtu-value*
10. **exit**
11. **interface port-channel** *channel-name*
12. **vlan-domain member** *vlan-domain-name*
13. **exit**
14. **interface port-channel** *channel-name.number*
15. **vrf member** *vrf-name* **tenant** *tenant-name*
16. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf switch or leaf switches to be configured. The <i>node-id</i> can be a single node ID or a range of IDs, in the form <i>node-id1-node-id2</i> , to which the configuration will be applied.
Step 3	vrf member <i>vrf-name</i> tenant <i>tenant-name</i> Example: <code>apic1(config-leaf-if)# vrf member v1 tenant t1</code>	Associates this port channel to this virtual routing and forwarding (VRF) instance and L3 outside policy, where:, where: • <i>vrf-name</i> is the VRF name. The name can be any case-sensitive, alphanumeric string up to 32 characters.

	Command or Action	Purpose
		<i>tenant-name</i> is the tenant name. The name can be any case-sensitive, alphanumeric string up to 32 characters.
Step 4	vlan-domain member <i>vlan-domain-name</i> Example: <code>apic1(config-leaf-if)# vlan-domain member dom1</code>	Associates the port channel template with the previously configured VLAN domain.
Step 5	ip address <i>ip-address / subnet-mask</i> Example: <code>apic1(config-leaf-if)# ip address 10.1.1.1/24</code>	Sets the IP address and subnet mask for the specified interface.
Step 6	ipv6 address <i>sub-bits/prefix-length preferred</i> Example: <code>apic1(config-leaf-if)# ipv6 address 2001::1/64 preferred</code>	Configures an IPv6 address based on an IPv6 general prefix and enables IPv6 processing on an interface, where: <i>sub-bits</i> is the subprefix bits and host bits of the address to be concatenated with the prefixes provided by the general prefix specified with the prefix-name argument. The sub-bits argument must be in the form documented in RFC 2373 where the address is specified in hexadecimal using 16-bit values between colons. <i>prefix-length</i> is the length of the IPv6 prefix. A decimal value that indicates how many of the high-order contiguous bits of the address comprise the prefix (the network portion of the address). A slash mark must precede the decimal value.
Step 7	ipv6 link-local <i>ipv6-link-local-address</i> Example: <code>apic1(config-leaf-if)# ipv6 link-local fe80::1</code>	Configures an IPv6 link-local address for an interface.
Step 8	mac-address <i>mac-address</i> Example: <code>apic1(config-leaf-if)# mac-address 00:44:55:66:55::01</code>	Manually sets the interface MAC address.
Step 9	mtu <i>mtu-value</i> Example: <code>apic1(config-leaf-if)# mtu 1500</code>	Sets the MTU for this class of service.
Step 10	exit Example: <code>apic1(config-leaf-if)# exit</code>	Returns to configure mode.

	Command or Action	Purpose
Step 11	interface port-channel <i>channel-name</i> Example: <code>apic1(config-leaf)# interface port-channel po1</code>	Enters the interface configuration mode for the specified port channel.
Step 12	vlan-domain member <i>vlan-domain-name</i> Example: <code>apic1(config-leaf-if)# vlan-domain member dom1</code>	Associates the port channel template with the previously configured VLAN domain.
Step 13	exit Example: <code>apic1(config-leaf-if)# exit</code>	Returns to configure mode.
Step 14	interface port-channel <i>channel-name.number</i> Example: <code>apic1(config-leaf)# interface port-channel po1.2001</code>	Enters the interface configuration mode for the specified sub-interface port channel.
Step 15	vrf member <i>vrf-name</i> tenant <i>tenant-name</i> Example: <code>apic1(config-leaf-if)# vrf member v1 tenant t1</code>	Associates this port channel to this virtual routing and forwarding (VRF) instance and L3 outside policy, where: where: • <i>vrf-name</i> is the VRF name. The name can be any case-sensitive, alphanumeric string up to 32 characters. • <i>tenant-name</i> is the tenant name. The name can be any case-sensitive, alphanumeric string up to 32 characters.
Step 16	exit Example: <code>apic1(config-leaf-if)# exit</code>	Returns to configure mode.

Example

This example shows how to configure a basic Layer 3 sub-interface port-channel.

```

apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface vlan 2001
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member v1 tenant t1
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# ip address 10.1.1.1/24
apic1(config-leaf-if)# ipv6 address 2001::1/64 preferred
apic1(config-leaf-if)# ipv6 link-local fe80::1
apic1(config-leaf-if)# mac-address 00:44:55:66:55::01
apic1(config-leaf-if)# mtu 1500

```

```

apic1(config-leaf-if)# exit
apic1(config-leaf)# interface port-channel po1
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface port-channel po1.2001
apic1(config-leaf-if)# vrf member vl tenant t1
apic1(config-leaf-if)# exit

```

Adding Ports to the Layer 3 Port-Channel Using the NX-OS CLI

This procedure adds ports to a Layer 3 port channel that you configured previously.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **interface Ethernet** *slot/port*
4. **channel-group** *channel-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	leaf <i>node-id</i> Example: apic1(config)# leaf 101	Specifies the leaf switch or leaf switches to be configured. The <i>node-id</i> can be a single node ID or a range of IDs, in the form <i>node-id1-node-id2</i> , to which the configuration will be applied.
Step 3	interface Ethernet <i>slot/port</i> Example: apic1(config-leaf)# interface Ethernet 1/1-2	Enters interface configuration mode for the interface you want to configure.
Step 4	channel-group <i>channel-name</i> Example: apic1(config-leaf-if)# channel-group p01	Configures the port in a channel group.

Example

This example shows how to add ports to a Layer 3 port-channel.

```

apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface Ethernet 1/1-2
apic1(config-leaf-if)# channel-group p01

```

Layer 3 Out to Layer 3 Out Inter-VRF Leaking

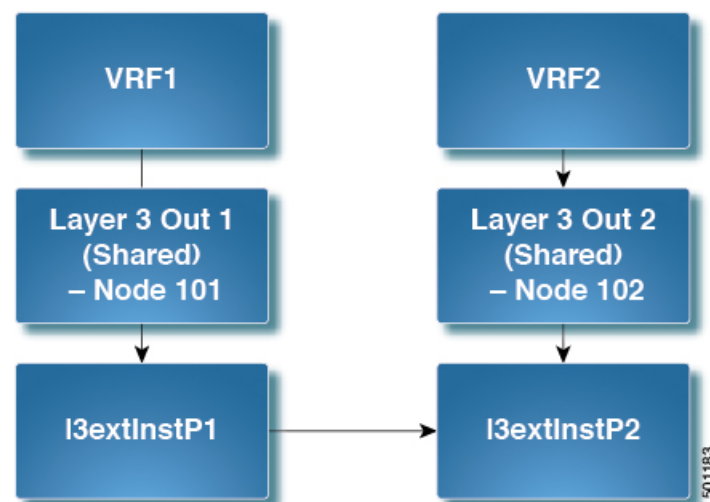
Starting with Cisco APIC release 2.2(2e), when there are two Layer 3 Outs in two different VRFs, inter-VRF leaking is supported.

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 18: Shared Layer 3 Outs Communicating Between Two VRFs



Configuring Shared Layer 3 Out Inter-VRF Leaking Using the NX-OS Style CLI - Named Example

SUMMARY STEPS

1. Enter the configure mode.
2. Configure the provider Layer 3 Out.
3. Configure the consumer Layer 3 Out.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Enter the configure mode. Example: apicl# configure	
Step 2	Configure the provider Layer 3 Out. Example: apicl(config)# tenant t1_provider apicl(config-tenant)# external-l3 epg l3extInstP-1 l3out T0-o1-L3OUT-1 apicl(config-tenant-l3ext-epg)# vrf member VRF1 apicl(config-tenant-l3ext-epg)# match ip 192.168.2.0/24 shared apicl(config-tenant-l3ext-epg)# contract provider vzBrCP-1 apicl(config-tenant-l3ext-epg)# exit apicl(config-tenant)# exit apicl(config)# leaf 101 apicl(config-leaf)# vrf context tenant t1_provider vrf VRF1 l3out T0-o1-L3OUT-1 apicl(config-leaf-vrf)# route-map T0-o1-L3OUT-1_shared apicl(config-leaf-vrf-route-map)# ip prefix-list l3extInstP-1 permit 192.168.2.0/24 apicl(config-leaf-vrf-route-map)# match prefix-list l3extInstP-1 apicl(config-leaf-vrf-route-map-match)# exit apicl(config-leaf-vrf-route-map)# exit apicl(config-leaf-vrf)# exit apicl(config-leaf)# exit	
Step 3	Configure the consumer Layer 3 Out. Example: apicl(config)# tenant t1_consumer apicl(config-tenant)# external-l3 epg l3extInstP-2 l3out T0-o1-L3OUT-1 apicl(config-tenant-l3ext-epg)# vrf member VRF2 apicl(config-tenant-l3ext-epg)# match ip 199.16.2.0/24 shared apicl(config-tenant-l3ext-epg)# contract consumer vzBrCP-1 imported apicl(config-tenant-l3ext-epg)# exit	

	Command or Action	Purpose
	<pre> apic1(config-tenant)# exit apic1(config)# leaf 101 apic1(config-leaf)# vrf context tenant t1_consumer vrf VRF2 l3out T0-o1-L3OUT-1 apic1(config-leaf-vrf)# route-map T0-o1-L3OUT-1_shared apic1(config-leaf-vrf-route-map)# ip prefix-list l3extInstP-2 permit 199.16.2.0/24 apic1(config-leaf-vrf-route-map)# match prefix-list l3extInstP-2 apic1(config-leaf-vrf-route-map-match)# exit apic1(config-leaf-vrf-route-map)# exit apic1(config-leaf-vrf)# exit apic1(config-leaf)# exit apic1(config)# </pre>	

Configuring Shared Layer 3 Out Inter-VRF Leaking Using the NX-OS Style CLI - Implicit Example

SUMMARY STEPS

1. Enter the configure mode.
2. Configure the provider tenant and VRF.
3. Configure the consumer tenant and VRF.
4. Configure the contract.
5. Configure the provider External Layer 3 EPG.
6. Configure the provider export map.
7. Configure the consumer external Layer 3 EPG.
8. Configure the consumer export map.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Enter the configure mode. Example: <pre>apic1# configure</pre>	
Step 2	Configure the provider tenant and VRF. Example: <pre> apic1(config)# tenant t1_provider apic1(config-tenant)# vrf context VRF1 apic1(config-tenant-vrf)# exit apic1(config-tenant)# exit </pre>	
Step 3	Configure the consumer tenant and VRF. Example:	

	Command or Action	Purpose
	<pre> apicl(config)# tenant t1_consumer apicl(config-tenant)# vrf context VRF2 apicl(config-tenant-vrf)# exit apicl(config-tenant)# exit </pre>	
Step 4	Configure the contract. Example: <pre> apicl(config)# tenant t1_provider apicl(config-tenant)# contract vzBrCP-1 type permit apicl(config-tenant-contract)# scope exportable apicl(config-tenant-contract)# export to tenant t1_consumer apicl(config-tenant-contract)# exit </pre>	
Step 5	Configure the provider External Layer 3 EPG. Example: <pre> apicl(config-tenant)# external-l3 epg l3extInstP-1 apicl(config-tenant-l3ext-epg)# vrf member VRF1 apicl(config-tenant-l3ext-epg)# match ip 192.168.2.0/24 shared apicl(config-tenant-l3ext-epg)# contract provider vzBrCP-1 apicl(config-tenant-l3ext-epg)# exit apicl(config-tenant)# exit </pre>	
Step 6	Configure the provider export map. Example: <pre> apicl(config)# leaf 101 apicl(config-leaf)# vrf context tenant t1_provider vrf VRF1 apicl(config-leaf-vrf)# route-map map1 apicl(config-leaf-vrf-route-map)# ip prefix-list p1 permit 192.168.2.0/24 apicl(config-leaf-vrf-route-map)# match prefix-list p1 apicl(config-leaf-vrf-route-map-match)# exit apicl(config-leaf-vrf-route-map)# exit apicl(config-leaf-vrf)# export map map1 apicl(config-leaf-vrf)# exit apicl(config-leaf)# exit </pre>	
Step 7	Configure the consumer external Layer 3 EPG. Example: <pre> apicl(config)# tenant t1_consumer apicl(config-tenant)# external-l3 epg l3extInstP-2 apicl(config-tenant-l3ext-epg)# vrf member VRF2 apicl(config-tenant-l3ext-epg)# match ip 199.16.2.0/24 shared apicl(config-tenant-l3ext-epg)# contract consumer vzBrCP-1 imported apicl(config-tenant-l3ext-epg)# exit apicl(config-tenant)# exit </pre>	
Step 8	Configure the consumer export map.	

Command or Action	Purpose
Example: <pre> apic1(config)# leaf 101 apic1(config-leaf)# vrf context tenant t1_consumer vrf VRF2 apic1(config-leaf-vrf)# route-map map2 apic1(config-leaf-vrf-route-map)# ip prefix-list p2 permit 199.16.2.0/24 apic1(config-leaf-vrf-route-map)# match prefix-list p2 apic1(config-leaf-vrf-route-map-match)# exit apic1(config-leaf-vrf-route-map)# exit apic1(config-leaf-vrf)# export map map2 apic1(config-leaf-vrf)# exit apic1(config-leaf)# exit apic1(config)# </pre>	

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 19: Local Scope Encapsulation and One Layer 3 Out

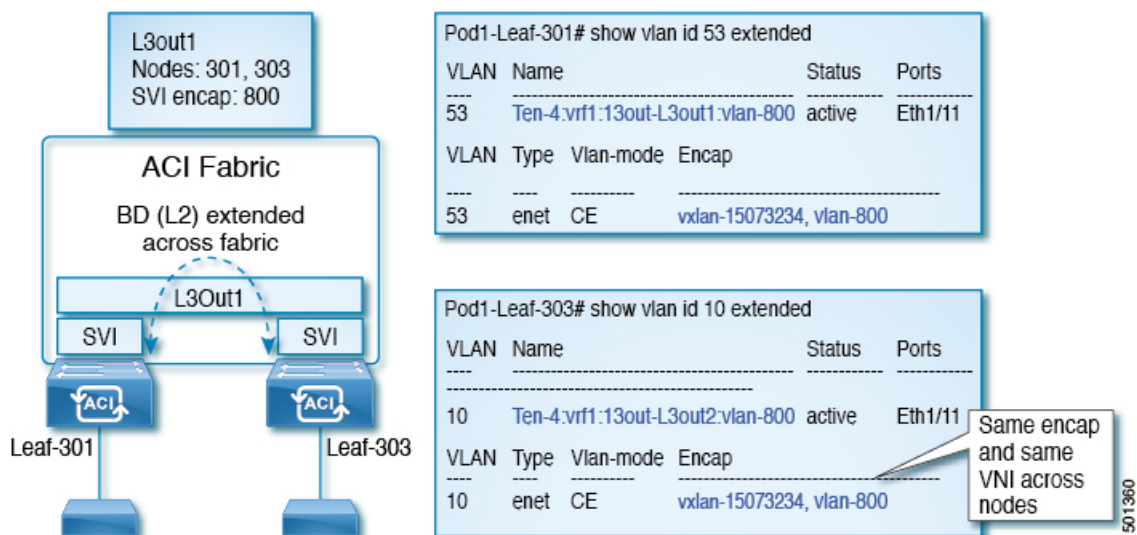
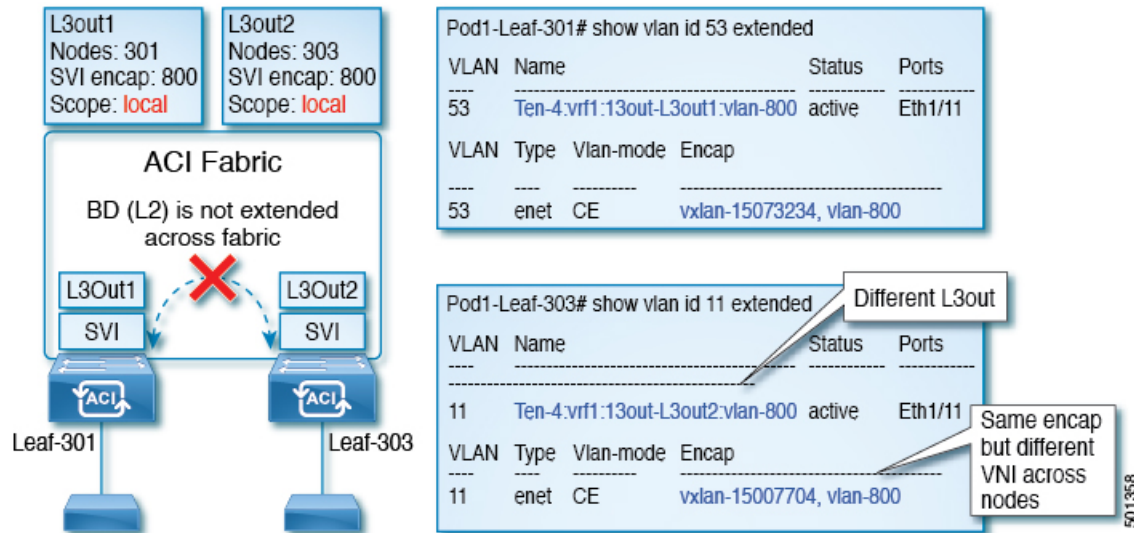


Figure 20: Local Scope Encapsulation and Two Layer 3 Outs

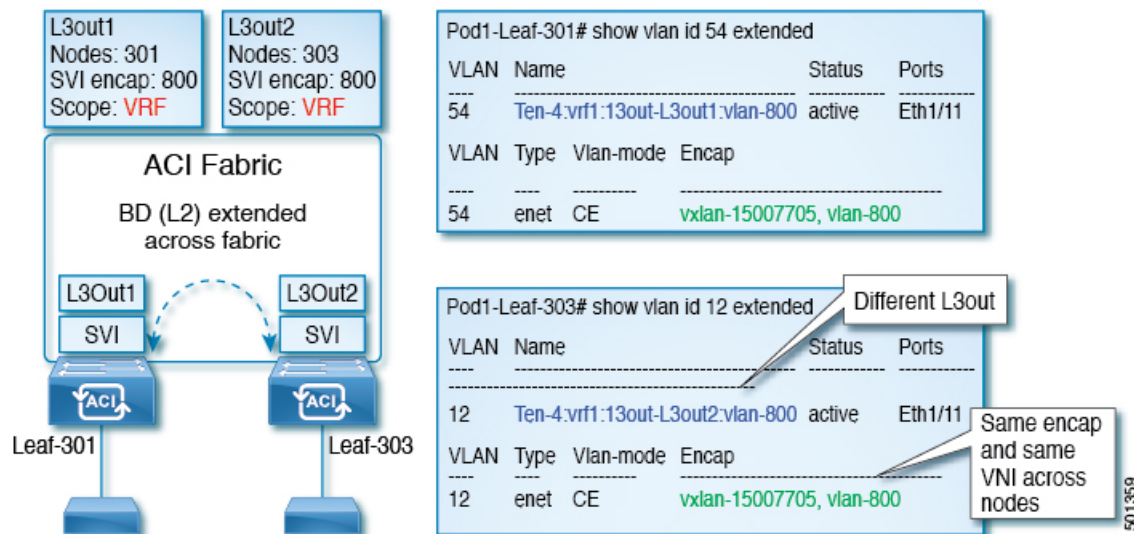


Starting with Cisco APIC release 2.3, it is now possible to choose the behavior when deploying two (or more) Layer 3 Outs using the same external encapsulation (SVI).

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 21: 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 10: 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 NX-OS Style CLI

The following example displaying steps for an SVI interface encapsulation scope setting is through a named Layer 3 Out configuration.

SUMMARY STEPS

1. Enter the configure mode.
2. Enter the switch mode.
3. Create the VLAN interface.
4. Specify the encapsulation scope.
5. Exit the interface mode.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Enter the configure mode. Example: <code>apic1# configure</code>	Enters the configuration mode.
Step 2	Enter the switch mode. Example: <code>apic1(config)# leaf 104</code>	Enters the switch mode.

	Command or Action	Purpose
Step 3	Create the VLAN interface. Example: <code>apicl(config-leaf)# interface vlan 2001</code>	Creates the VLAN interface. The VLAN range is 1-4094.
Step 4	Specify the encapsulation scope. Example: <code>apicl(config-leaf-if)# encap scope vrf context</code>	Specifies the encapsulation scope.
Step 5	Exit the interface mode. Example: <code>apicl(config-leaf-if)# exit</code>	Exits the interface mode.

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 11: 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 NX-OS Style CLI

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.

SUMMARY STEPS

1. Enter the configure mode.
2. Enter the switch mode.
3. Create the VLAN interface.
4. Enable SVI auto state.
5. Exit the interface mode.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Enter the configure mode. Example: <code>apic1# configure</code>	Enters the configuration mode.
Step 2	Enter the switch mode. Example: <code>apic1(config)# leaf 104</code>	Enters the switch mode.
Step 3	Create the VLAN interface. Example: <code>apic1(config-leaf)# interface vlan 2001</code>	Creates the VLAN interface. The VLAN range is 1-4094.
Step 4	Enable SVI auto state. Example: <code>apic1(config-leaf-if)# autostate</code>	Enables SVI auto state. By default, the SVI auto state value is not enabled.
Step 5	Exit the interface mode. Example: <code>apic1(config-leaf-if)# exit</code>	Exits the interface mode.

Configuring an Interface and Static Route

Before you begin

Configure a tenant and VRF.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **[no] vrf context tenant** *tenant-name* **vrf** *vrf-name*
4. (Optional) **[no] router-id** *ipv4-address*
5. **[no] {ip | ipv6} route** *ip-prefix/masklen* *next-hop-address* **[preferred]**
6. **exit**
7. **interface** *type*
8. **vlan-domain member** *domain-name*
9. **no switchport**
10. **vrf member tenant** *tenant-name* **vrf** *vrf-name*
11. **[no] {ip | ipv6} address** *ip-prefix/masklen* **[eui64] [secondary] [preferred]**
12. **[[no]]ip dhcprelayaddress** *tenant**tenant-name* *dhcp-address* **{application***app-name*
*epg**epg-name***|external-12***12-epg-name***|external-13***13-epg-name***}**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	[no] vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-leaf)# vrf context tenant exampleCorp vrf v1</code>	Configures a tenant VRF on the node.
Step 4	(Optional) [no] router-id <i>ipv4-address</i> Example: <code>apic1(config-leaf-vrf)# router-id 1.2.3.4</code>	Assigns a router ID for routing protocols running on the VRF. If you do not assign a router ID, an ID is generated internally that is unique to each leaf switch.
Step 5	[no] {ip ipv6} route <i>ip-prefix/masklen</i> <i>next-hop-address</i> [preferred] Example:	Configures static route information for the VRF.

	Command or Action	Purpose
	<pre>apic1(config-leaf-vrf) # ip route 21.1.1.1/32 32.1.1.1 apic1(config-leaf-vrf) # ipv6 route 5001::1/128 6002::1</pre>	
Step 6	exit Example: <pre>apic1(config-leaf-vrf) # exit</pre>	Returns to leaf configuration mode.
Step 7	interface type Example: <pre>apic1(config-leaf) # interface eth 1/1</pre>	Specifies a port for the external interface.
Step 8	vlan-domain member domain-name Example: <pre>apic1(config-leaf-if) # vlan-domain member dom1</pre>	Assign a VLAN domain to the interface. The VLAN domain must have already been created using the vlan-domain command in the global configuration mode.
Step 9	no switchport Example: <pre>apic1(config-leaf-if) # no switchport</pre>	Configures the interface as a layer 3 interface, exposing the layer 3 commands in the configuration options.
Step 10	vrf member tenant tenant-name vrf vrf-name Example: <pre>apic1(config-leaf-if) # vrf member tenant exampleCorp vrf v1</pre>	Attaches the interface to the tenant VRF.
Step 11	[no] {ip ipv6} address ip-prefix/masklen [eui64] [secondary] [preferred] Example: <pre>apic1(config-leaf-if) # ip address 10.1.1.1/24 apic1(config-leaf-if) # ipv6 address 2001::1/64 preferred</pre>	Configures IP addresses on the interface. The specified address can be declared as either: • preferred—The default source address for traffic from the interface. • secondary—The secondary address of the interface. With the optional eui64 keyword, the host can assign itself a 64-bit Extended Unique Identifier (EUI). In this mode, you can also configure ipv6 link-local, mac address, mtu, and other layer 3 properties on the interface.
Step 12	[[no]]ip dhcprelayaddress tenanttenant-name dhcp-address {applicationapp-name epgepg-name external-12 2-epg-name external-13 3-epg-name} Example: <pre>apic1(config-leaf-if) # ip dhcp relay address 192.0.20.1 tenant exampleCorp application app1 epg epg1</pre>	Sets or removes a DHCP relay address for the external interface along with any supported options.

Examples

This example shows how to deploy a layer 3 port for external connectivity.

```
apicl# configure
apicl(config)# leaf 101
apicl(config-leaf)# vrf context tenant exampleCorp vrf v1
apicl(config-leaf-vrf)# router-id 1.2.3.4
apicl(config-leaf-vrf)# ip route 21.1.1.1/32 32.1.1.1
apicl(config-leaf-vrf)# ipv6 route 5001::1/128 6002::1 preferred
apicl(config-leaf-vrf)# exit

apicl(config-leaf)# interface eth 1/1
apicl(config-leaf-if)# vlan-domain member dom1
apicl(config-leaf-if)# no switchport
apicl(config-leaf-if)# vrf member tenant exampleCorp vrf v1
apicl(config-leaf-if)# ip address 10.1.1.1/24
apicl(config-leaf-if)# ip address 11.1.1.1/24 secondary
apicl(config-leaf-if)# ipv6 address 2001::1/64 preferred
apicl(config-leaf-if)# ipv6 link-local fe80::1
apicl(config-leaf-if)# mac-address 00:44:55:66:55::01
apicl(config-leaf-if)# mtu 4470
```

This example shows how to configure a layer 3 subinterface port for external connectivity. In this example, the subinterface ID (the "100" in 1/2.100) is actually the VLAN encapsulation instead of an ID. All properties supported on a layer 3 port are available on the subinterface as well.

```
apicl# configure
apicl(config)# leaf 101
    # SAME VRF CONTEXT CONFIGURATION AS PREVIOUS EXAMPLE

apicl(config-leaf)# interface eth 1/2.100
apicl(config-leaf-if)# vrf member tenant exampleCorp vrf v1
    # SAME L3 PROPERTIES CONFIGURATION AS PREVIOUS EXAMPLE
```

This example shows the methods to configure a switched virtual interface (SVI) for external connectivity. Each external SVI is uniquely identified by its encapsulation VLAN denoted in the SVI ID.

```
apicl# configure
apicl(config)# leaf 101
    # SAME VRF CONTEXT CONFIGURATION AS PREVIOUS EXAMPLE

apicl(config-leaf)# interface vlan 200
apicl(config-leaf-if)# vrf member tenant exampleCorp vrf v1
apicl(config-leaf-if)# ip address 13.1.1.1/24

    # HOW TO ATTACH A PORT TO THE EXTERNAL SVI:
apicl(config)# leaf 101
apicl(config-leaf)# interface eth 1/4
apicl(config-leaf-if)# vlan-domain member dom1
apicl(config-leaf)# switchport trunk allowed vlan 10 tenant exampleCorp external-svi

    # HOW TO ATTACH A PORT CHANNEL TO THE EXTERNAL SVI:
apicl(config)# leaf 102
apicl(config-leaf)# interface port-channel po1
apicl(config-leaf-if)# vlan-domain member dom1
apicl(config-leaf)# switchport trunk allowed vlan 10 tenant exampleCorp external-svi
```



```
# HOW TO ATTACH A VIRTUAL PORT CHANNEL (vPC) TO THE EXTERNAL SVI:
apic1(config)# vpc context leaf 101 102
apic1(config-leaf)# interface vpc vpc103
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf)# switchport trunk allowed vlan 10 tenant exampleCorp external-svi
```

**Note**

An external SVI must be configured on each of the participating nodes. This allows you to configure different IP addresses on each of the nodes for the same SVI. If the vPC is part of an external SVI, you must individually create an SVI on each of the participating vPC peers and you can provide different IP addresses on each SVI.

OSPF Configuration

Configuring OSPF

OSPF can operate in one of the following modes in an area:

- When OSPF is used as the main routing protocol for the tenant VRF in the node, OSPF will import and export routes defined in the route-map configured in the OSPF area. The route-map contains the export routes.
- When OSPF is used as a connectivity protocol for BGP, OSPF advertises the loopback address which is used as the source of the BGP session. Note that the loopback IP address and not the loopback ID is used. In this case, a BGP session relying on OSPF will use the same loopback IP address in its update-source command.

There is no need for separate configuration of OSPF and OSPFv3. The router OSPF mode handles both OSPFv2 and OSPFv3 implicitly for the areas running under OSPF.

OSPF sessions are supported on all types of layer 3 Interfaces in the leaf, including:

- Layer 3 ports
- Subinterfaces
- External SVI

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **router ospf default**
4. **vrf member tenant** *tenant-name* **vrf** *vrf-name*
5. (Optional) **default-information originate** [**always**]
6. **area** *area-id* **nssa** [**no-redistribution**] [**default-information-originate**]
7. **area** *area-id* **stub**

8. **area** *area-id* **default-cost** *cost*
9. **area** *area-id* **route-map** *map-name* **out**
10. **area** *area-id* **loopback** *loopback-address*
11. **inherit** {**ipv4** | **ipv6**} **ospf vrf-policy** *policy-name*
12. **summary-address** *ip-address*
13. **area** *area-id* **range** *address-range* **cost** *cost*
14. **exit**
15. **exit**
16. **interface** *slot/port*
17. {**ip** | **ipv6**} **router ospf default area** *area-id*
18. {**ip** | **ipv6**} **ospf inherit interface-policy** *if-policy-name* **tenant** *tenant-name*
19. [**no**] {**ip** | **ipv6**} **ospf prefix-suppression** {**enable** | **disable** | **inherit**}
20. [**no**] {**ip** | **ipv6**} **ospf passive-interface**
21. [**no**] **ip ospf authentication** {**md5** | **none** | **simple**}
22. **ip ospf authentication-key** *key*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	router ospf default Example: <code>apic1(config-leaf)# router ospf default</code>	Creates an OSPF routing process and enters OSPF policy configuration.
Step 4	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-leaf-ospf)# vrf member tenant exampleCorp vrf v100</code>	Enables a VRF in the OSPF session.
Step 5	(Optional) default-information originate [always] Example: <code>apic1(config-leaf-ospf-vrf)# default-information originate</code>	Causes the switch to generate a default route.
Step 6	area <i>area-id</i> nssa [no-redistribution] [default-information-originate] Example: <code>apic1(config-leaf-ospf-vrf)# area 0 nssa</code>	Defines a not-so-stubby area (NSSA).

	Command or Action	Purpose
Step 7	area <i>area-id</i> stub Example: <code>apicl(config-leaf-ospf-vrf) # area 17 stub</code>	Defines an area to be a stub area.
Step 8	area <i>area-id</i> default-cost <i>cost</i> Example: <code>apicl(config-leaf-ospf-vrf) # area 17 default-cost 20</code>	Sets OSPF default area cost to a value between 0 and 16777215.
Step 9	area <i>area-id</i> route-map <i>map-name</i> out Example: <code>apicl(config-leaf-ospf-vrf) # area 17 route-map ospf-to-eigrp out</code>	Specifies a route-map for outbound filtering.
Step 10	area <i>area-id</i> loopback <i>loopback-address</i> Example: <code>apicl(config-leaf-ospf-vrf) # area 17 loopback 192.0.20.11/32</code>	When OSPF is used as a connectivity protocol for BGP, OSPF advertises the loopback address which is used as the source of the BGP session. Note that the loopback IP address and not the loopback ID is used. In this case, a BGP session relying on OSPF will use the same loopback IP address in its update-source command.
Step 11	inherit {ipv4 ipv6} ospf vrf-policy <i>policy-name</i> Example: <code>apicl(config-leaf-ospf-vrf) # inherit ipv4 ospf vrf-policy vrfTemplate2</code>	Inherits the OSPF Template Policy under this VRF.
Step 12	summary-address <i>ip-address</i> Example: <code>apicl(config-leaf-ospf-vrf) # summary-address 182.1.20.0/24</code>	Configures external route summarization. Enter the summary address for external routes learned from other protocols.
Step 13	area <i>area-id</i> range <i>address-range</i> cost <i>cost</i> Example: <code>apicl(config-leaf-ospf-vrf) # area 17 range 192.0.20.0/24 cost 20</code>	Configures inter-area route summarization, which summarizes the networks between areas. The range is the summary route to be advertised in areas. The cost is a value between 0 and 16777215.
Step 14	exit Example: <code>apicl(config-leaf-ospf-vrf) # exit</code>	Returns to OSPF configuration mode.
Step 15	exit Example: <code>apicl(config-leaf-ospf) # exit</code>	Returns to leaf configuration mode.

	Command or Action	Purpose
Step 16	interface <i>slot/port</i> Example: apic1(config-leaf) # interface eth 1/2	Specifies a port for the OSPF interface. The interface could also be specified as interface <i>slot/port.vlan-id</i> or interface <i>vlanvlan-id</i> .
Step 17	{ip ipv6} router ospf default area <i>area-id</i> Example: apic1(config-leaf-if) # ip router ospf default area 17	Creates an OSPF routing process and enters OSPF policy configuration.
Step 18	{ip ipv6} ospf inherit interface-policy <i>if-policy-name</i> tenant <i>tenant-name</i> Example: apic1(config-leaf-if) # ip ospf inherit interface-policy ifPolicy3 tenant exampleCorp	Inherits the OSPF interface template policy under this tenant.
Step 19	[no] {ip ipv6} ospf prefix-suppression {enable disable inherit} Example: apic1(config-leaf-if) # ip ospf prefix-suppression enable	Prevents OSPF from advertising all IP prefixes that belong to a specific interface, except for prefixes that are associated with secondary IP addresses.
Step 20	[no] {ip ipv6} ospf passive-interface Example: apic1(config-leaf-if) # ip ospf passive-interface	Suppresses routing updates on the interface.
Step 21	[no] ip ospf authentication {md5 none simple} Example: apic1(config-leaf-if) # ip ospf authentication md5	Specifies the authentication type.
Step 22	ip ospf authentication-key <i>key</i> Example: apic1(config-leaf-if) # ip ospf authentication-key c1\$c0123	Specifies the authentication key.

Examples

```

apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant exampleCorp vrf v100
apic1(config-leaf-ospf-vrf)# area 0 nssa
apic1(config-leaf-ospf-vrf)# area 17 stub
apic1(config-leaf-ospf-vrf)# area 17 default-cost 20
apic1(config-leaf-ospf-vrf)# area 17 route-map ospf-to-eigrp out
apic1(config-leaf-ospf-vrf)# area 17 loopback 192.0.20.11/32
apic1(config-leaf-ospf-vrf)# inherit ipv4 ospf vrf-policy vrfTemplate2
apic1(config-leaf-ospf-vrf)# summary-address 182.1.20.0/24

```

```

apic1(config-leaf-ospf-vrf)# area 17 range 192.0.20.0/24 cost 20
apic1(config-leaf-ospf-vrf)# exit
apic1(config-leaf-ospf)# exit
apic1(config-leaf)# interface eth 1/3
apic1(config-leaf-if)# ip router ospf default area 17
apic1(config-leaf-if)# ip ospf inherit interface-policy ifPolicy3 tenant exampleCorp
apic1(config-leaf-if)# ip ospf prefix-suppression enable
apic1(config-leaf-if)# ip ospf passive-interface
apic1(config-leaf-if)# ip ospf authentication md5
apic1(config-leaf-if)# ip ospf authentication-key c1$c0123

```

Creating OSPF VRF and Interface Templates

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **template ospf vrf-policy** *vrf-policy-name* **tenant** *tenant-name*
4. **timers throttle lsa** *start-time* *hold-interval* *max-time*
5. **timers lsa-group-pacing** *seconds*
6. **timers lsa-arrival** *milliseconds*
7. **timers throttle spf** *spf-start* *spf-hold* *spf-max-wait*
8. **auto-cost reference-bandwidth** *bandwidth*
9. **distance** *distance*
10. **maximum-paths** *max-paths*
11. **graceful-restart** *helper-disable*
12. **prefix-suppression**
13. **name-lookup**
14. **exit**
15. **template ospf interface-policy** *if-policy-name* **tenant** *tenant-name*
16. **[no] advertise-subnet**
17. **[no] cost** *if-cost*
18. **[no] dead-interval** *seconds*
19. **[no] hello-interval** *seconds*
20. **[no] mtu-ignore**
21. **[no] network** {*bcast* | *p2p* | *unspecified*}
22. **[no] passive-interface**
23. **[no] priority** *priority*
24. **[no] retransmit-interval** *seconds*
25. **[no] transmit-delay** *seconds*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example:	Enters configuration mode.

	Command or Action	Purpose
	<code>apic1# configure</code>	
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	template ospf vrf-policy <i>vrf-policy-name</i> tenant <i>tenant-name</i> Example: <code>apic1(config-leaf)# template ospf vrf-policy vrfTemplate3 tenant exampleCorp</code>	Creates the OSPF VRF policy template under the specified tenant.
Step 4	timers throttle lsa <i>start-time</i> <i>hold-interval</i> <i>max-time</i> Example: <code>apic1(config-vrf-policy)# timers throttle lsa 200 10000 45000</code>	Sets the start-interval, hold-interval, and max-interval for link-state advertisements (LSA).
Step 5	timers lsa-group-pacing <i>seconds</i> Example: <code>apic1(config-vrf-policy)# timers lsa-group-pacing 240</code>	Sets the interval in which LSAs are grouped and refreshed, checksummed, or aged.
Step 6	timers lsa-arrival <i>milliseconds</i> Example: <code>apic1(config-vrf-policy)# timers lsa-arrival 1000</code>	Sets the minimum interval between the arrival of each LSA.
Step 7	timers throttle spf <i>spf-start</i> <i>spf-hold</i> <i>spf-max-wait</i> Example: <code>apic1(config-vrf-policy)# timers throttle spf 5 1000 90000</code>	Sets the SPF init-interval, hold-interval, and max-interval for LSA.
Step 8	auto-cost reference-bandwidth <i>bandwidth</i> Example: <code>apic1(config-vrf-policy)# auto-cost reference-bandwidth 1000</code>	Sets OSPF Policy Bandwidth Reference in Mbps.
Step 9	distance <i>distance</i> Example: <code>apic1(config-vrf-policy)# distance 200</code>	Sets OSPF Policy Preferred Administrative Distance.
Step 10	maximum-paths <i>max-paths</i> Example: <code>apic1(config-vrf-policy)# maximum-paths 8</code>	Sets the maximum number of parallel routes that OSPF can install in a routing table. The range is from 1 to 16 routes.
Step 11	graceful-restart helper-disable Example:	Disables the graceful restart helper mode.

	Command or Action	Purpose
	<code>apic1(config-vrf-policy)# graceful-restart helper-disable</code>	
Step 12	prefix-suppression Example: <code>apic1(config-vrf-policy)# prefix-suppression</code>	Prevents OSPF from advertising all IP prefixes except prefixes that are associated with loopbacks, secondary IP addresses, and passive interfaces.
Step 13	name-lookup Example: <code>apic1(config-vrf-policy)# name-lookup</code>	Configures OSPF to look up DNS names.
Step 14	exit Example: <code>apic1(config-vrf-policy)# exit</code>	Returns to leaf configuration mode.
Step 15	template ospf interface-policy if-policy-name tenant tenant-name Example: <code>apic1(config-leaf)# template ospf interface-policy ifTemplate5 tenant exampleCorp</code>	Creates the OSPF interface policy template under the specified tenant.
Step 16	[no] advertise-subnet Example: <code>apic1(config-interface-policy)# advertise-subnet</code>	Advertises the primary IP address subnet mask instead of /32.
Step 17	[no] cost if-cost Example: <code>apic1(config-interface-policy)# cost 300</code>	Sets the OSPF cost for the interface. The range is 0 to 65535.
Step 18	[no] dead-interval seconds Example: <code>apic1(config-interface-policy)# dead-interval 60</code>	Sets the interval in seconds at which hello packets must not be seen before neighbors declare the router down. The range is 1 to 65535 seconds.
Step 19	[no] hello-interval seconds Example: <code>apic1(config-interface-policy)# hello-interval 10</code>	Specifies the interval between hello packets in seconds. The range is 1 to 65535 seconds.
Step 20	[no] mtu-ignore Example: <code>apic1(config-interface-policy)# mtu-ignore</code>	Disables MTU mismatch detection on the interface.
Step 21	[no] network {bcast p2p unspecified} Example: <code>apic1(config-interface-policy)# network p2p</code>	Sets the OSPF interface policy network type, which can be broadcast or point-to-point.

	Command or Action	Purpose
Step 22	[no] passive-interface Example: apicl(config-interface-policy) # passive-interface	Suppresses OSPF routing updates on the interface.
Step 23	[no] priority <i>priority</i> Example: apicl(config-interface-policy) # priority 4	Sets OSPF interface priority, which is used to determine the designated router (DR) on a specific network. The range is 0 to 255.
Step 24	[no] retransmit-interval <i>seconds</i> Example: apicl(config-interface-policy) # retransmit-interval 5	Specifies the time between link-state advertisement (LSA) retransmissions for adjacencies belonging to the interface. The range is 1 to 65535 seconds.
Step 25	[no] transmit-delay <i>seconds</i> Example: apicl(config-interface-policy) # transmit-delay 2	Sets the estimated time required to send a link-state update packet on the interface. The range is from 1 to 450 seconds.

Examples

This example shows how to configure a VRF template and an interface template.

```

apicl# configure
apicl(config)# leaf 101

# CONFIGURING THE VRF TEMPLATE:
apicl(config-leaf)# template ospf vrf-policy vrfTemplate3 tenant exampleCorp
apicl(config-vrf-policy)# timers throttle lsa 200 10000 45000
apicl(config-vrf-policy)# timers lsa-group-pacing 240
apicl(config-vrf-policy)# timers lsa-arrival 1000
apicl(config-vrf-policy)# timers throttle spf 5 1000 90000
apicl(config-vrf-policy)# auto-cost reference-bandwidth 1000
apicl(config-vrf-policy)# distance 200
apicl(config-vrf-policy)# maximum-paths 8
apicl(config-vrf-policy)# graceful-restart helper-disable
apicl(config-vrf-policy)# prefix-suppression
apicl(config-vrf-policy)# name-lookup
apicl(config-vrf-policy)# exit

# CONFIGURING THE INTERFACE TEMPLATE:
apicl(config-leaf)# template ospf interface-policy ifTemplate5 tenant exampleCorp
apicl(config-ospf-if-policy)# advertise-subnet
apicl(config-ospf-if-policy)# cost 300
apicl(config-ospf-if-policy)# dead-interval 60
apicl(config-ospf-if-policy)# hello-interval 10
apicl(config-ospf-if-policy)# mtu-ignore
apicl(config-ospf-if-policy)# network p2p
apicl(config-ospf-if-policy)# passive-interface
apicl(config-ospf-if-policy)# priority 4
apicl(config-ospf-if-policy)# retransmit-interval 5
apicl(config-ospf-if-policy)# transmit-delay 2

```


BGP Configuration

Configuring BGP

SUMMARY STEPS

1. **configure**
2. **bgp-fabric**
3. **asn** *asn-number*
4. **route-reflector** *spine spine-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	bgp-fabric Example: <code>apic1(config)# bgp-fabric</code>	Enters BGP configuration mode for the fabric.
Step 3	asn <i>asn-number</i> Example: <code>apic1(config-bgp-fabric)# asn 100</code>	Specifies the BGP autonomous system number (ASN).
Step 4	route-reflector <i>spine spine-id</i> Example: <code>apic1(config-bgp-fabric)# route-reflector spine 105</code>	Configures the specified spine switch to be a BGP route reflector.

Examples

```
apic1# configure
apic1(config)# bgp-fabric
apic1(config-bgp-fabric)# asn 100
apic1(config-bgp-fabric)# route-reflector spine 105
```

What to do next

Configure BGP address family and counters.

Creating BGP Address Family and Timer Templates

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **template bgp timers** *timer-policy-name* **tenant** *tenant-name*
4. **graceful-restart-helper**
5. **graceful-restart stalepath-time** *seconds*
6. **timers bgp** *keep-alive-seconds* *hold-seconds*
7. **exit**
8. **template bgp address-family** *family-name* **tenant** *tenant-name*
9. **distance** *ebgp-distance* *ibgp-distance* *local-distance*
10. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	leaf <i>node-id</i> Example: apic1(config)# leaf 101	Specifies the leaf to be configured.
Step 3	template bgp timers <i>timer-policy-name</i> tenant <i>tenant-name</i> Example: apic1(config-leaf)# template bgp timers bgpTimers tenant exampleCorp This template will be available on all leaves where tenant exampleCorp has a VRF deployment	Creates the BGP timers policy template under the specified tenant.
Step 4	graceful-restart-helper	Configure BGP Policy Graceful Restart Helper apic1(config-bgp-timers)# graceful-restart-helper
Step 5	graceful-restart stalepath-time <i>seconds</i>	Sets the maximum time that BGP keeps stale routes from the restarting BGP peer. The range is 1 to 3600 seconds. apic1(config-bgp-timers)# graceful-restart stalepath-time 3600
Step 6	timers bgp <i>keep-alive-seconds</i> <i>hold-seconds</i>	Sets the keep-alive timer and hold timer values. The range for both is 1 to 3600 seconds. apic1(config-bgp-timers)# timers bgp 10 20
Step 7	exit	apic1(config-bgp-timers)# exit

	Command or Action	Purpose
Step 8	template bgp address-family <i>family-name</i> tenant <i>tenant-name</i> Example: <pre>apic1(config-leaf)# template bgp address-family bgpAf1 tenant exampleCorp This template will be available on all leaves where tenant exampleCorp has a VRF deployment</pre>	Creates the BGP address family template under the specified tenant.
Step 9	distance <i>ebgp-distance</i> <i>ibgp-distance</i> <i>local-distance</i> <pre>apic1(config-bgp-af)# distance 250 240 230</pre>	Sets the administrative distance for eBGP routes, iBGP routes, and local routes. The range is 1 to 255.
Step 10	exit <pre>apic1(config-bgp-af)# exit</pre>	Returns to leaf configuration mode.

Examples

This example shows how to create a BGP timer template and an address family template.

```
apic1# configure
apic1(config)# leaf 101

# CREATE A TIMER TEMPLATE
apic1(config-leaf)# template bgp timers bgpTimers tenant exampleCorp
This template will be available on all leaves where tenant exampleCorp has a VRF deployment
apic1(config-bgp-timers)# timers bgp 10 20
apic1(config-bgp-timers)# graceful-restart stalepath-time 3600
apic1(config-bgp-timers)# exit

# CREATE AN ADDRESS FAMILY TEMPLATE
apic1(config-leaf)# template bgp address-family bgpAf1 tenant bgp_t1
This template will be available on all leaves where tenant exampleCorp has a VRF deployment
apic1(config-bgp-af)# distance 250 240 230
apic1(config-bgp-af)# exit
apic1(config-leaf)# exit
```

Configuring BGP Address Family and Timers

Before you begin

Create a BGP address family template and timer template.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **router bgp** *asn-number*
4. **vrf member tenant** *tenant-name* **vrf** *vrf-name*

5. **inherit bgp timer** *timer-name*
6. **address-family** {*ipv4* | *ipv6*} **unicast**
7. **inherit bgp address-family** *family-name*
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: apicl(config)# leaf 101	Specifies the leaf to be configured.
Step 3	router bgp <i>asn-number</i> Example: apicl(config-leaf)# router bgp 100	Enters BGP policy configuration.
Step 4	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: apicl(config-bgp)# vrf member tenant exampleCorp vrf v100	Specifies the VRF instance to associate with subsequent address family configuration mode commands.
Step 5	inherit bgp timer <i>timer-name</i> Example: apicl(config-leaf-bgp-vrf)# inherit bgp timer bgpTimers This template will be inherited on all leaves where VRF v100 has been deployed	Applies an existing timer configuration.
Step 6	address-family { <i>ipv4</i> <i>ipv6</i> } unicast Example: apicl(config-leaf-bgp-vrf)# address-family ipv4 unicast	Declares neighbors with whom we want to exchange normal IPv4 unicast routes.
Step 7	inherit bgp address-family <i>family-name</i> Example: apicl(config-leaf-bgp-vrf-af)# inherit bgp address-family ipv4-af-pol This template will be inherited on all leaves where VRF v100 has been deployed	Adds the specified address family to this address family.
Step 8	exit Example: apicl(config-leaf-bgp-vrf-af)# exit	

Examples

This example shows how to inherit a BGP timer configuration and IPv4 and IPv6 address families.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# router bgp 100
apic1(config-bgp)# vrf member tenant exampleCorp vrf v100
apic1(config-leaf-bgp-vrf)# inherit bgp timer bgpTimers
This template will be inherited on all leaves where VRF v100 has been deployed
apic1(config-leaf-bgp-vrf)# address-family ipv4 unicast
apic1(config-leaf-bgp-vrf-af)# inherit bgp address-family ipv4-af-pol
This template will be inherited on all leaves where VRF v100 has been deployed
apic1(config-leaf-bgp-vrf-af)# exit
apic1(config-leaf-bgp-vrf)# address-family ipv6 unicast
apic1(config-leaf-bgp-vrf-af)# inherit bgp address-family ipv6-af-pol
This template will be inherited on all leaves where VRF v100 has been deployed
apic1(config-leaf-bgp-vrf-af)# exit
apic1(config-leaf-bgp-vrf)# exit
apic1(config-leaf)# exit
```

Configuring a BGP Neighbor

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **router bgp** *asn-number*
4. **vrf member tenant** *tenant-name* **vrf** *vrf-name*
5. (Optional) **aggregate-address** *ip-address/masklength* [**as-set**]
6. **neighbor** *ip-address* [/masklength]
7. **address-family** {**ipv4** | **ipv6**} **unicast**
8. [**no**] **maximum-prefix** *count* [**action** {**log** | **shut** | **restart** [*restart-timeminutes*]}] [**threshold** *percent*]
9. **exit**
10. **update-source** {**loopback** *ip-address* | **ethernet** *ip-address* | **vlan** *vlan-id*}
11. **weight** *number*
12. **private-as-control** {**remove-exclusive** | **remove-exclusive-all** | **remove-exclusive-all-replace-as**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example:	Specifies the leaf to be configured.

	Command or Action	Purpose
	<code>apic1(config)# leaf 101</code>	
Step 3	router bgp <i>asn-number</i> Example: <code>apic1(config-leaf)# router bgp 100</code>	Enters BGP policy configuration.
Step 4	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-bgp)# vrf member tenant exampleCorp vrf v100</code>	Specifies the VRF instance to associate with subsequent policy configuration mode commands.
Step 5	(Optional) aggregate-address <i>ip-address/masklength</i> [as-set] Example: <code>apic1(config-leaf-bgp-vrf)# aggregate-address 192.0.10.0/24 as-set</code>	Configures a summary address for a range of addresses and creates an aggregate entry in a BGP database. The address can be either IPv4 or IPv6. The as-set option generates autonomous system set path information.
Step 6	neighbor <i>ip-address</i>[/masklength] Example: <code>apic1(config-leaf-bgp-vrf)# neighbor 192.0.2.229/32</code>	Specifies the IP address of the neighbor.
Step 7	address-family {ipv4 ipv6} unicast Example: <code>apic1(config-leaf-bgp-vrf-neighbor)# address-family ipv4 unicast</code>	Declares neighbors with whom we want to exchange normal IPv4 unicast routes.
Step 8	[no] maximum-prefix <i>count</i> [action {log shut restart} [restart-time <i>minutes</i>]] [threshold <i>percent</i>] Example: <code>apic1(config-leaf-bgp-vrf-neighbor-af)# maximum-prefix 10 threshold 10 action restart restart-time 10</code>	Sets the maximum number of prefixes from this neighbor. the range is 1 to 300000 prefixes. Other optional settings are: • action— The action to be performed when the maximum prefix limit is reached. If the action is restart, you can optionally specify the restart-time, which is the period of time in minutes before restarting the peer when the maximum prefix limit is reached. The range is 1 to 65535 minutes. • threshold— The threshold percentage of the maximum number of prefixes before a warning is issued. The range is 1 to 100 percent.
Step 9	exit Example: <code>apic1(config-leaf-bgp-vrf-neighbor-af)# exit</code>	

	Command or Action	Purpose
Step 10	update-source { <i>loopback ip-address</i> <i>ethernet ip-address</i> <i>vlan vlan-id</i> } Example: <pre>apic1(config-leaf-bgp-vrf-neighbor) # update-source loopback 192.0.2.230</pre>	if the neighbor address is being learned through OSPF, specify the same loopback address as being used under OSPF.
Step 11	weight <i>number</i> Example: <pre>apic1(config-leaf-bgp-vrf-neighbor) # weight 2000</pre>	Specifies the weight attribute to select a best path. A weight can from 0 to 65,535. Routes with a higher weight value have preference when there are multiple routes to the same destination.
Step 12	private-as-control { remove-exclusive remove-exclusive-all remove-exclusive-all-replace-as } Example: <pre>apic1(config-leaf-bgp-vrf-neighbor) # private-as-control</pre>	Removes private autonomous system numbers from the autonomous system path. Private AS numbers can be removed from the AS path on a per peer basis and can only be used for eBGP peers according to the following three possible variations: • remove-exclusive—Remove if AS path has only private AS numbers. • remove-exclusive-all—Remove if AS path has both private and public AS numbers. • remove-exclusive-all-replace-as—Replaces private AS numbers with the router's local AS number. This command is shown as an example. At this point you can configure any of the neighbor settings shown in the table below.

The following table shows the interface settings that can be configured at this point.

Command	Purpose
allow-self-as	Accept as-path with my AS present in it
allowed-self-as-count <i>count</i>	The number of occurrences of a local access service network
disable-connected-check	Disable check for directly connected peer
disable-peer-as-check	Disable checking of peer AS-number while advertising
ebgp-multihop <i>count</i>	Specify multihop TTL for remote peer
local-as <i>asn</i>	Local Autonomous System Configuration for a BGP Peer
next-hop-self	Set our peering address as nexthop
password <i>password</i>	Configure a password for neighbor

Command	Purpose
<code>private-as-control</code>	Removes private ASNs from the AS path
<code>remote-as <i>asn</i></code>	Specify Autonomous System Number of the neighbor
<code>route-map <i>name</i> {in out}</code>	Apply route-map to neighbor
<code>send-community [extended]</code>	Send Community attribute to this neighbor
<code>update-source vlan <i>vlan-id</i></code>	Source Vlan Interface
<code>update-source ethernet <i>slot/port</i></code>	Source Ethernet Interface
<code>update-source loopback <i>ip-address</i></code>	Source Loopback Interface
<code>weight <i>number</i></code>	BGP weight for the routing table

Examples

This example shows how to configure an IPv4 BGP neighbor.

```

apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# router bgp 100
apic1(config-bgp)# vrf member tenant exampleCorp vrf v100
apic1(config-leaf-bgp-vrf)# aggregate-address 192.0.10.0/24 as-set
apic1(config-leaf-bgp-vrf)# neighbor 192.0.2.229/32
apic1(config-leaf-bgp-vrf-neighbor)# address-family ipv4 unicast
apic1(config-leaf-bgp-vrf-neighbor-af)# maximum-prefix 10 threshold 10 action restart
restart-time 10
apic1(config-leaf-bgp-vrf-neighbor-af)# exit
apic1(config-leaf-bgp-vrf-neighbor)# allow-self-as
apic1(config-leaf-bgp-vrf-neighbor)# allowed-self-as-count 2
apic1(config-leaf-bgp-vrf-neighbor)# disable-connected-check
apic1(config-leaf-bgp-vrf-neighbor)# disable-peer-as-check
apic1(config-leaf-bgp-vrf-neighbor)# ebgp-multihop 4
apic1(config-leaf-bgp-vrf-neighbor)# local-as 100
apic1(config-leaf-bgp-vrf-neighbor)# next-hop-self
apic1(config-leaf-bgp-vrf-neighbor)# password abcdef
apic1(config-leaf-bgp-vrf-neighbor)# remote-as 200
apic1(config-leaf-bgp-vrf-neighbor)# send-community extended
apic1(config-leaf-bgp-vrf-neighbor)# update-source vlan 601
apic1(config-leaf-bgp-vrf-neighbor)# update-source ethernet 1/15
apic1(config-leaf-bgp-vrf-neighbor)# update-source loopback 192.0.2.230
Warning: BGP Configuration changed. Please re-configure BGP Password if it was enabled
apic1(config-leaf-bgp-vrf-neighbor)# local-as 100 no-prepend replace-as dual-as
apic1(config-leaf-bgp-vrf-neighbor)# route-map rMapT3 out
apic1(config-leaf-bgp-vrf-neighbor)# weight 2000
apic1(config-leaf-bgp-vrf-neighbor)# private-as-control
apic1(config-leaf-bgp-vrf-neighbor)# exit
apic1(config-leaf-bgp-vrf)# exit
apic1(config-leaf)# exit

```

This example shows how to configure an IPv6 BGP neighbor.

```

apic1# configure

```



```

apic1(config)# leaf 101
apic1(config-leaf)# router bgp 100
apic1(config-bgp)# vrf member tenant exampleCorp vrf v100
apic1(config-leaf-bgp-vrf)# neighbor 2001:80:1:2::229
apic1(config-leaf-bgp-vrf-neighbor)# address-family ipv6 unicast
apic1(config-leaf-bgp-vrf-neighbor-af)# maximum-prefix 100
apic1(config-leaf-bgp-vrf-neighbor-af)# exit
apic1(config-leaf-bgp-vrf-neighbor)# allow-self-as
apic1(config-leaf-bgp-vrf-neighbor)# allowed-self-as-count 2
apic1(config-leaf-bgp-vrf-neighbor)# disable-connected-check
apic1(config-leaf-bgp-vrf-neighbor)# disable-peer-as-check
apic1(config-leaf-bgp-vrf-neighbor)# ebgp-multihop 4
apic1(config-leaf-bgp-vrf-neighbor)# local-as 100
apic1(config-leaf-bgp-vrf-neighbor)# next-hop-self
apic1(config-leaf-bgp-vrf-neighbor)# password abcdef
apic1(config-leaf-bgp-vrf-neighbor)# remote-as 200
apic1(config-leaf-bgp-vrf-neighbor)# send-community extended
apic1(config-leaf-bgp-vrf-neighbor)# update-source vlan 601
apic1(config-leaf-bgp-vrf-neighbor)# update-source ethernet 1/15
apic1(config-leaf-bgp-vrf-neighbor)# update-source loopback 2001:80:1:2::230/128
Warning: BGP Configuration changed. Please re-configure BGP Password if it was enabled
apic1(config-leaf-bgp-vrf-neighbor)# local-as 100 no-prepend replace-as dual-as
apic1(config-leaf-bgp-vrf-neighbor)# route-map rMapT3 out
apic1(config-leaf-bgp-vrf-neighbor)# weight 2000
apic1(config-leaf-bgp-vrf-neighbor)# private-as-control
apic1(config-leaf-bgp-vrf-neighbor)# exit
apic1(config-leaf-bgp-vrf-af)# exit
apic1(config-leaf-bgp-vrf)# exit
apic1(config-leaf)# exit

```

Configuring a Per VRF Per Node BGP Timer Policy Using the NX-OS Style CLI

SUMMARY STEPS

1. Configure BGP ASN and the route reflector before creating a timer policy.
2. Create a timer policy.
3. Display the configured BGP policy.
4. Refer to a specific policy at a node.
5. Display the node specific BGP timer policy.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Configure BGP ASN and the route reflector before creating a timer policy. Example: <pre> apic1(config)# apic1(config)# bgp-fabric apic1(config-bgp-fabric)# route-reflector spine 102 apic1(config-bgp-fabric)# asn 42 apic1(config-bgp-fabric)# exit </pre>	

	Command or Action	Purpose
	<pre>apic1(config)# exit apic1#</pre>	
Step 2	Create a timer policy. Example: <pre>apic1# config apic1(config)# leaf 101 apic1(config-leaf)# template bgp timers pol7 tenant tn1 This template will be available on all nodes where tenant tn1 has a VRF deployment apic1(config-bgp-timers)# timers bgp 120 240 apic1(config-bgp-timers)# graceful-restart stalepath-time 500 apic1(config-bgp-timers)# maxas-limit 300 apic1(config-bgp-timers)# exit apic1(config-leaf)# exit apic1(config)# exit apic1#</pre>	The specific values are provided as examples only.
Step 3	Display the configured BGP policy. Example: <pre>apic1# show run leaf 101 template bgp timers pol7 # Command: show running-config leaf 101 template bgp timers pol7 leaf 101 template bgp timers pol7 tenant tn1 timers bgp 120 240 graceful-restart stalepath-time 500 maxas-limit 300 exit exit</pre>	
Step 4	Refer to a specific policy at a node. Example: <pre>apic1# config apic1(config)# leaf 101 apic1(config-leaf)# router bgp 42 apic1(config-leaf-bgp)# vrf member tenant tn1 vrf ctx1 apic1(config-leaf-bgp-vrf)# inherit node-only bgp timer pol7 apic1(config-leaf-bgp-vrf)# exit apic1(config-leaf-bgp)# exit apic1(config-leaf)# exit apic1(config)# exit apic1#</pre>	
Step 5	Display the node specific BGP timer policy. Example: <pre>apic1# show run leaf 101 router bgp 42 vrf member</pre>	

Command or Action	Purpose
<pre> tenant tn1 vrf ctx1 # Command: show running-config leaf 101 router bgp 42 vrf member tenant tn1 vrf ctx1 leaf 101 router bgp 42 vrf member tenant tn1 vrf ctx1 inherit node-only bgp timer pol7 exit exit exit apic1# </pre>	

Configuring BGP Max Path

Before you begin

The appropriate tenant and the BGP external routed network are created and available.

The following feature enables you to add the maximum number of paths to the route table to enable equal cost, multipath load balancing.

The two properties which enable you to configure more paths are maxEcmp and maxEcmpIbgp in the bgpCtxAfPol object. After you configure these two properties, they are propagated to the rest of your implementation.

Use the following commands when logged in to BGP:

```

maximum-paths [ibgp]
no maximum-paths [ibgp]

```

Example Configuration:

SUMMARY STEPS

- 1.

DETAILED STEPS

Example:

```

apic1(config)# leaf 101
apic1(config-leaf)# template bgp address-family newAf tenant t1
This template will be available on all nodes where tenant t1 has a VRF deployment
apic1(config-bgp-af)# maximum-paths ?
<1-16> Maximum number of equal-cost paths for load sharing. The default is 16.
ibgp  Configure multipath for IBGP paths
apic1(config-bgp-af)# maximum-paths 10
apic1(config-bgp-af)# maximum-paths ibgp 8
apic1(config-bgp-af)# end
apic1#
no maximum-paths [ibgp]

```

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 • 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 NX-OS Style CLI

This section provides information on how to configure the AS Path Prepend feature using the NX-OS style command line interface (CLI).

Before you begin

A configured tenant.

SUMMARY STEPS

1. To modify the autonomous system path (AS Path) for Border Gateway Protocol (BGP) routes, you can use the `set as-path` command. The `set as-path` command takes the form of

```
apic1(config-leaf-vrf-template-route-profile)# set
as-path {'prepend as-num [ ,... as-num ] | prepend-last-as num}
```

DETAILED STEPS

To modify the autonomous system path (AS Path) for Border Gateway Protocol (BGP) routes, you can use the `set as-path` command. The `set as-path` command takes the form of `apic1(config-leaf-vrf-template-route-profile)# set as-path {'prepend as-num [ ,... as-num ] | prepend-last-as num}`

Example:

```
apic1(config)# leaf 103
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# template route-profile rp1
apic1(config-leaf-vrf-template-route-profile)# set as-path ?
prepend Prepend to the AS-Path
prepend-last-as Prepend last AS to the as-path
apic1(config-leaf-vrf-template-route-profile)# set as-path prepend 100, 101, 102, 103
apic1(config-leaf-vrf-template-route-profile)# set as-path prepend-last-as 8
apic1(config-leaf-vrf-template-route-profile)# exit
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# exit
```

What to do next

To disable AS Path prepend, use the no form of the shown command:

```
apic1(config-leaf-vrf-template-route-profile)# [no] set
as-path { prepend as-num [ ,... as-num ] | prepend-last-as num }
```

Route Distribution Into BGP

Configuring a Route-Profile with Tenant Scope

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **template route-profile** *profile-name* **tenant** *tenant-name*
4. **[no] set tag** *name*
5. **exit**
6. **template route-profile** *profile-name* **tenant** *tenant-name*
7. **[no] set tag** *name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	leaf <i>node-id</i> Example:	Specifies the leaf to be configured.

	Command or Action	Purpose
	<code>apicl(config)# leaf 101</code>	
Step 3	template route-profile <i>profile-name</i> tenant <i>tenant-name</i> Example: <code>apicl(config-leaf)# template route-profile map_eigrp tenant exampleCorp</code>	Creates a route-profile template under tenant for BGP dampening and route redistribution.
Step 4	Required: [no] set tag name Example: <code>apicl(config-leaf-template-route-profile)# set tag 200</code>	Sets the tag value. The <i>name</i> parameter is an unsigned integer.
Step 5	Required: exit Example: <code>apicl(config-leaf-template-route-profile)# exit</code>	Returns to leaf configuration mode.
Step 6	template route-profile <i>profile-name</i> tenant <i>tenant-name</i> Example: <code>apicl(config-leaf)# template route-profile map_ospf tenant exampleCorp</code>	Creates a route-profile template under tenant for BGP dampening and route redistribution.
Step 7	Required: [no] set tag name Example: <code>apicl(config-leaf-template-route-profile)# set tag 100</code>	Sets the tag value. The <i>name</i> parameter is an unsigned integer.

Example

```

apicl# configure
apicl(config)# leaf 101
apicl(config-leaf)# template route-profile map_eigrp tenant exampleCorp
apicl(config-leaf-template-route-profile)# set tag 200
apicl(config-leaf-template-route-profile)# exit
apicl(config-leaf)# template route-profile map_ospf tenant exampleCorp
apicl(config-leaf-template-route-profile)# set tag 100
apicl(config-leaf-template-route-profile)# exit

```

What to do next

Configure a redistribute route-profile under BGP for OSPF and EIGRP using one of the route-profiles created in this procedure.

Configuring a Redistribute Route-Profile

Before you begin

Create a route-profile template under tenant for route redistribution.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **router bgp** *asn-number*
4. **vrf member tenant** *tenant-name* **vrf** *vrf-name*
5. **redistribute {ospf | eigrp} route-map** *map-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	router bgp <i>asn-number</i> Example: <code>apic1(config-leaf)# router bgp 100</code>	Enters BGP policy configuration.
Step 4	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-bgp)# vrf member tenant exampleCorp vrf v100</code>	Specifies the VRF instance to associate with subsequent policy configuration mode commands.
Step 5	Required: redistribute {ospf eigrp} route-map <i>map-name</i> Example: <code>apic1(config-leaf-bgp-vrf)# redistribute ospf route-map map_ospf</code>	

Example

This example configures a redistribute route-profile under BGP for OSPF and EIGRP using the route-profiles created in the example in *Creating a Route-Profile with Tenant Scope*. The redistribute route-map allows (permits) all routes and applies the route-profile for the route-control actions. In this example, all EIGRP learned routes will be redistributed into BGP with tag 200 and OSPF routes will be redistributed into BGP with tag 100.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# router bgp 100
apic1(config-bgp)# vrf member tenant exampleCorp vrf v1
apic1(config-leaf-bgp-vrf)# redistribute eigrp route-map map_eigrp
apic1(config-leaf-bgp-vrf)# redistribute ospf route-map map_ospf
```

Configuring BGP Route Dampening

BGP route dampening minimizes propagation into the fabric of flapping eBGP routes received from external routers connected to border leaf switches (BLs). Frequently flapping routes from external routers are suppressed on BLs based on configured criteria and prohibited from redistribution to iBGP peers (ACI spine switches). Suppressed routes are reused after a configured time criteria. Each flap penalizes the eBGP route with a penalty of 1000. When the flap penalty reaches a defined suppress-limit threshold (default 2000) the eBGP route is marked as dampened. Dampened routes are not advertised to other BGP peers. The penalty is decremented to half after every half-life interval (the default is 15 minutes). A dampened route is reused if the penalty falls below a specified reuse-limit (the default is 750). A dampened route is suppressed at most for a specified maximum suppress time (maximum of 45 minutes).

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **template route-profile** *profile-name* **tenant** *tenant-name*
4. **[no] set dampening** *half-life* *reuse* *suppress* *max-suppress-time*
5. **exit**
6. **router bgp** *asn-number*
7. **vrf member** **tenant** *tenant-name* **vrf** *vrf-name*
8. **neighbor** *ip-address*[/*masklength*]
9. **address-family** {*ipv4* | *ipv6*} **unicast**
10. **inherit bgp dampening** *profile-name*
11. **exit**
12. **exit**
13. **address-family** {*ipv4* | *ipv6*} **unicast**
14. **inherit bgp dampening** *profile-name*
15. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	template route-profile <i>profile-name</i> tenant <i>tenant-name</i> Example: <code>apic1(config-leaf)# template route-profile damp_rp tenant exampleCorp</code>	Creates a route-profile template under tenant for BGP dampening and route redistribution.

	Command or Action	Purpose
Step 4	Required: [no] set dampening <i>half-life reuse suppress max-suppress-time</i> Example: <pre>apic1(config-leaf-template-route-profile)# set dampening 15 750 2000 60</pre>	Configures route flap dampening behavior. The parameters are: • <i>half-life</i>—Decay half life, which is the time in minutes after which a penalty is decreased. Once the route has been assigned a penalty, the penalty is decreased by half after the half life period. The range is 1 to 60 minutes; the default is 15 minutes. • <i>reuse</i>—A route is unsuppressed (reused) if the penalty for a flapping route decreases enough to fall below this value. The range is 1 to 20000; the default is 750. • <i>suppress</i>—A route is suppressed when its penalty exceeds this limit. The range is 1 to 20000; the default is 2000. • <i>max-suppress-time</i>—The maximum time in minutes that a stable route can be suppressed. The range is 1 to 255.
Step 5	Required: exit Example: <pre>apic1(config-leaf-template-route-profile)# exit</pre>	Returns to leaf configuration mode.
Step 6	router bgp <i>asn-number</i> Example: <pre>apic1(config-leaf)# router bgp 100</pre>	Enters BGP policy configuration.
Step 7	vrf member tenant <i>tenant-name vrf vrf-name</i> Example: <pre>apic1(config-bgp)# vrf member tenant exampleCorp vrf v100</pre>	Specifies the VRF instance to associate with subsequent policy configuration mode commands.
Step 8	neighbor <i>ip-address[/masklength]</i> Example: <pre>apic1(config-leaf-bgp-vrf)# neighbor 192.0.2.229/32</pre>	Specifies the IP address of the neighbor. The mask length must be 32.
Step 9	address-family {<i>ipv4 ipv6</i>} unicast Example: <pre>apic1(config-leaf-bgp-vrf-neighbor)# address-family ipv4 unicast</pre>	Declares neighbors with whom we want to exchange normal IPv4 unicast routes.
Step 10	inherit bgp dampening <i>profile-name</i> Example: <pre>apic1(config-leaf-bgp-vrf-neighbor-af)# inherit bgp dampening damp_rp</pre>	

	Command or Action	Purpose
Step 11	exit Example: <code>apic1(config-leaf-bgp-vrf-neighbor-af) # exit</code>	
Step 12	exit Example: <code>apic1(config-leaf-bgp-vrf-neighbor) # exit</code>	
Step 13	address-family {ipv4 ipv6} unicast Example: <code>apic1(config-leaf-bgp-vrf) # address-family ipv4 unicast</code>	Declares neighbors with whom we want to exchange normal IPv4 unicast routes.
Step 14	inherit bgp dampening <i>profile-name</i> Example: <code>apic1(config-leaf-bgp-vrf-af) # inherit bgp dampening damp_rp</code>	
Step 15	exit Example: <code>apic1(config-leaf-bgp-vrf-af) # exit</code>	

Example

```

apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# template route-profile damp_rp tenant exampleCorp
apic1(config-leaf-template-route-profile)# set dampening 15 750 2000 60
apic1(config-leaf-template-route-profile)# exit
apic1(config-leaf)# router bgp 100
apic1(config-bgp)# vrf member tenant exampleCorp vrf v100
apic1(config-leaf-bgp-vrf)# neighbor 192.0.2.229/32
apic1(config-leaf-bgp-vrf-neighbor)# address-family ipv4 unicast
apic1(config-leaf-bgp-vrf-neighbor-af)# inherit bgp dampening damp_rp
apic1(config-leaf-bgp-vrf-neighbor-af)# exit
apic1(config-leaf-bgp-vrf)# address-family ipv6 unicast
apic1(config-leaf-bgp-vrf-af)# inherit bgp dampening damp_rp
apic1(config-leaf-bgp-vrf-af)# exit

```

EIGRP Configuration

Creating EIGRP VRF and Interface Templates

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **template eigrp vrf-policy** *vrf-policy-name* **tenant** *tenant-name*
4. **distance** *internal external*
5. **maximum-paths** *limit*
6. **metric version** *64bit*
7. **timers active-time** *minutes*
8. **template eigrp interface-policy** *if-policy-name* **tenant** *tenant-name*
9. **ip hello-interval eigrp default** *seconds*
10. **ip hold-interval eigrp default** *seconds*
11. **ip next-hop-self eigrp default**
12. **ip passive-interface eigrp default**
13. **ip split-horizon eigrp default**
14. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	template eigrp vrf-policy <i>vrf-policy-name</i> tenant <i>tenant-name</i> Example: <code>apic1(config-leaf)# template eigrp vrf-policy vrfTemplate3 tenant exampleCorp</code> This template will be available on all leaves where tenant exampleCorp has a VRF deployment	Creates the EIGRP VRF policy template under the specified tenant.
Step 4	distance <i>internal external</i> Example: <code>apic1(config-template-eigrp-vrf-pol)# distance 2 5</code>	Sets EIGRP administrative distance preference for internal and external routes. The distances can be 1 to 255.

	Command or Action	Purpose
Step 5	maximum-paths <i>limit</i> Example: <pre>apic1(config-template-eigrp-vrf-pol)# maximum-paths 8</pre>	Sets EIGRP Maximum Path Limit for the VRF policy template. The limit can be 1 to 32.
Step 6	metric version 64bit Example: <pre>apic1(config-template-eigrp-vrf-pol)# metric version 64bit</pre>	Sets EIGRP metric style to wide metric (64 bits).
Step 7	timers active-time <i>minutes</i> Example: <pre>apic1(config-template-eigrp-vrf-pol)# timers active-time 1</pre>	Sets EIGRP active timer interval. The range is 1 to 65535 minutes.
Step 8	template eigrp interface-policy <i>if-policy-name</i> tenant <i>tenant-name</i> Example: <pre>apic1(config-leaf)# template eigrp interface-policy ifTemplate5 tenant exampleCorp</pre> This template will be available on all leaves where tenant exampleCorp has a VRF deployment	Creates the EIGRP interface policy template under the specified tenant.
Step 9	ip hello-interval eigrp default <i>seconds</i> Example: <pre>apic1(config-template-eigrp-if-pol)# ip hello-interval eigrp default 10</pre>	Sets EIGRP hello interval time. The range is 1 to 65535 seconds.
Step 10	ip hold-interval eigrp default <i>seconds</i> Example: <pre>apic1(config-template-eigrp-if-pol)# ip hold-interval eigrp default 10</pre>	Sets EIGRP hold interval time. The range is 1 to 65535 seconds.
Step 11	ip next-hop-self eigrp default Example: <pre>apic1(config-template-eigrp-if-pol)# ip next-hop-self eigrp default</pre>	Sets EIGRP next-hop-self flag.
Step 12	ip passive-interface eigrp default Example: <pre>apic1(config-template-eigrp-if-pol)# ip passive-interface eigrp default</pre>	Set EIGRP passive-interface flag.
Step 13	ip split-horizon eigrp default Example: <pre>apic1(config-template-eigrp-if-pol)# ip split-horizon eigrp default</pre>	Sets EIGRP split-horizon flag.

	Command or Action	Purpose
Step 14	exit Example: <code>apic1(config-template-eigrp-if-pol)# exit</code>	Returns to leaf configuration mode.

Examples

```

apic1# configure
apic1(config)# leaf 101

# CONFIGURING THE VRF TEMPLATE:
apic1(config-leaf)# template eigrp vrf-policy vrfTemplate3 tenant exampleCorp
This template will be available on all leaves where tenant exampleCorp has a VRF deployment
apic1(config-template-eigrp-vrf-pol)# distance 2 5
apic1(config-template-eigrp-vrf-pol)# maximum-paths 8
apic1(config-template-eigrp-vrf-pol)# metric version 64bit
apic1(config-template-eigrp-vrf-pol)# timers active-time 1
apic1(config-template-eigrp-vrf-pol)# exit

# CONFIGURING THE INTERFACE TEMPLATE:
apic1(config-leaf)# template eigrp interface-policy ifTemplate5 tenant exampleCorp
This template will be available on all leaves where tenant exampleCorp has a VRF deployment
apic1(config-template-eigrp-if-pol)# ip hello-interval eigrp default 5
apic1(config-template-eigrp-if-pol)# ip hold-interval eigrp default 10
apic1(config-template-eigrp-if-pol)# ip next-hop-self eigrp default
apic1(config-template-eigrp-if-pol)# ip passive-interface eigrp default
apic1(config-template-eigrp-if-pol)# ip split-horizon eigrp default
apic1(config-template-eigrp-if-pol)# exit
apic1(config-leaf)# exit
apic1(config)# exit

```

What to do next

Configure EIGRP address family and counters.

Configuring EIGRP Address Family and Counters

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **router eigrp default**
4. **vrf member tenant** *tenant-name* **vrf** *vrf-name*
5. **autonomous-system** *asn*
6. **address-family** {*ipv4* | *ipv6*} **unicast**
7. **distance** *internal* *external*
8. **maximum-paths** *limit*
9. **metric version** *64bit*
10. **timers active-time** *minutes*

11. inherit eigrp vrf-policy vrf-policy-name**DETAILED STEPS**

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	leaf node-id Example: apic1(config)# leaf 101	Specifies the leaf to be configured.
Step 3	router eigrp default Example: apic1(config-leaf)# router eigrp default	Enters EIGRP policy configuration.
Step 4	vrf member tenant tenant-name vrf vrf-name Example: apic1(config-eigrp)# vrf member tenant exampleCorp vrf v100	Specifies the VRF instance to associate with subsequent address family configuration mode commands.
Step 5	autonomous-system asn Example: apic1(config-eigrp-vrf)# autonomous-system 300	Enters Autonomous System configuration for EIGRP.
Step 6	address-family {ipv4 ipv6} unicast Example: apic1(config-eigrp-vrf)# address-family ipv4 unicast	Configures an EIGRP policy address family.
Step 7	distance internal external Example: apic1(config-address-family)# distance 2 5	Sets EIGRP administrative distance preference for internal and external routes. The distances can be 1 to 255.
Step 8	maximum-paths limit Example: apic1(config-address-family)# maximum-paths 8	Sets EIGRP Maximum Path Limit for the VRF policy template. The limit can be 1 to 32.
Step 9	metric version 64bit Example: apic1(config-address-family)# metric version 64bit	Sets EIGRP metric style to wide metric (64 bits).
Step 10	timers active-time minutes Example:	Sets EIGRP active timer interval. The range is 1 to 65535 minutes.

	Command or Action	Purpose
	apic1(config-address-family)# timers active-time 1	
Step 11	inherit eigrp vrf-policy vrf-policy-name Example: apic1(config-address-family)# inherit eigrp vrf-policy vrfTemplate3	Applies an EIGRP VRF policy to this address family.

Examples

This example shows how to configure an EIGRP address family and inherit an EIGRP VRF policy.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# router eigrp default
apic1(config-eigrp)# vrf member tenant exampleCorp vrf v100
apic1(config-eigrp-vrf)# autonomous-system 300
apic1(config-eigrp-vrf)# address-family ipv4 unicast
This configuration will affect all leaves where VRF v100 has been deployed
apic1(config-address-family)# distance 2 5
This configuration will affect all leaves where VRF v100 has been deployed
apic1(config-address-family)# maximum-paths 8
This configuration will affect all leaves where VRF v100 has been deployed
apic1(config-address-family)# metric version 64bit
This configuration will affect all leaves where VRF v100 has been deployed
apic1(config-address-family)# timers active-time 1
This configuration will affect all leaves where VRF v100 has been deployed
apic1(config-address-family)# inherit eigrp vrf-policy vrfTemplate3
This template will be inherited on all leaves where VRF v100 has been deployed
apic1(config-address-family)# exit
apic1(config-eigrp-vrf)# exit
apic1(config-eigrp)# exit
```

Configuring an EIGRP Interface

SUMMARY STEPS

1. **configure**
2. **leaf node-id**
3. **interface ethernet slot/port**
4. **[no] switchport slot/port**
5. **[no] vlan-domain member vlan-id**
6. **[no] vrf member tenant exampleCorp vrf vrf-name**
7. **[no] {ip | ipv6} address ip-address/mask-length**
8. **[no] {ip | ipv6} router eigrp default**
9. **[no] {ip | ipv6} distribute-list eigrp default route-map map-name out**
10. **[no] {ip | ipv6} hello-interval eigrp default seconds**
11. **[no] {ip | ipv6} hold-interval eigrp default seconds**

12. [no] {ip | ipv6} next-hop-self eigrp default
13. [no] {ip | ipv6} passive-interface eigrp default
14. [no] {ip | ipv6} split-horizon eigrp default
15. [no] inherit eigrp ip interface-policy *if-policy-name*
16. [no] ip summary-address eigrp default *ip-prefix*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: apic1(config)# leaf 101	Specifies the leaf to be configured.
Step 3	interface ethernet <i>slot/port</i> Example: apic1(config-leaf)# interface ethernet 1/21	Specifies the interface to be configured.
Step 4	[no] switchport <i>slot/port</i> Example: apic1(config-leaf-if)# no switchport	By default, a port is in Layer 2 trunk mode. If the port is in Layer 3 mode, it must be converted to Layer 2 trunk mode using this command.
Step 5	[no] vlan-domain member <i>vlan-id</i> Example: apic1(config-leaf-if)# vlan-domain member dom1	Creates and enters the configuration mode for the VLAN domain.
Step 6	[no] vrf member tenant <i>exampleCorp</i> vrf <i>vrf-name</i> Example: apic1(config-leaf-if)# vrf member tenant exampleCorp vrf v100	Associates the interface with a VRF.
Step 7	[no] {ip ipv6} address <i>ip-address/mask-length</i> Example: apic1(config-leaf-if)# ip address 181.12.12.1/24	Sets an IP address for the interface.
Step 8	[no] {ip ipv6} router eigrp default Example: apic1(config-leaf-if)# ip router eigrp default	Sets router EIGRP policies to default.
Step 9	[no] {ip ipv6} distribute-list eigrp default route-map <i>map-name</i> out Example:	EIGRP advertises routes that are matched in the route-map specified in the distribute-list command. The route prefixes mentioned in the prefix-list in the route-map can be learned from other protocol sources like BGP, OSPF, Static,

	Command or Action	Purpose
	<code>apic1(config-leaf-if)# ip distribute-list eigrp default route-map rMapT5 out</code>	Connected. Redistribute route-maps are automatically created based on the distribute-list command. Note that prefixes learned from an EIGRP session running on another interface on the same switch will not be filtered by the distribute-list and will always be advertised out.
Step 10	[no] {ip ipv6} hello-interval eigrp default <i>seconds</i> Example: <code>apic1(config-leaf-if)# ip hello-interval eigrp default 10</code>	Sets EIGRP hello interval time. The range is 1 to 65535 seconds.
Step 11	[no] {ip ipv6} hold-interval eigrp default <i>seconds</i> Example: <code>apic1(config-leaf-if)# ip hold-interval eigrp default 10</code>	Sets EIGRP hold interval time. The range is 1 to 65535 seconds.
Step 12	[no] {ip ipv6} next-hop-self eigrp default Example: <code>apic1(config-leaf-if)# ip next-hop-self eigrp default</code>	Sets EIGRP next-hop-self flag.
Step 13	[no] {ip ipv6} passive-interface eigrp default Example: <code>apic1(config-leaf-if)# ip passive-interface eigrp default</code>	Set EIGRP passive-interface flag.
Step 14	[no] {ip ipv6} split-horizon eigrp default Example: <code>apic1(config-leaf-if)# ip split-horizon eigrp default</code>	Sets EIGRP split-horizon flag.
Step 15	[no] inherit eigrp ip interface-policy <i>if-policy-name</i> Example: <code>apic1(config-leaf-if)# inherit eigrp ip interface-policy ifTemplate5</code>	Applies an EIGRP interface policy to this interface.
Step 16	[no] ip summary-address eigrp default <i>ip-prefix</i> Example: <code>apic1(config-leaf-if)# ip summary-address eigrp default 172.10.1.0/24</code> <code>apic1(config-leaf-if)# ip summary-address eigrp default 2001::/64</code>	Configures route summarization for EIGRP. A summary address can be configured to advertise an aggregated prefix on an EIGRP session. Note A summary address enabled on one interface will also be applied on other EIGRP enabled interfaces on the same VRF on the switch.

Examples

This example shows how to configure an EIGRP interface.

```

apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/21
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# vrf member tenant exampleCorp vrf v100
apic1(config-leaf-if)# ip address 181.12.12.1/24
apic1(config-leaf-if)# ip router eigrp default
apic1(config-leaf-if)# ip distribute-list eigrp default route-map rMapT5 out
distribute list will be updated on all EIGRP interfaces on node 1021 VRF exampleCorp/v100
apic1(config-leaf-if)# ip hello-interval eigrp default 5
apic1(config-leaf-if)# ip hold-interval eigrp default 10
apic1(config-leaf-if)# ip next-hop-self eigrp default
apic1(config-leaf-if)# ip passive-interface eigrp default
apic1(config-leaf-if)# ip split-horizon eigrp default
apic1(config-leaf-if)# inherit eigrp ip interface-policy ifTemplate5
apic1(config-leaf-if)# ip summary-address eigrp default 172.10.1.0/24
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# exit

```

Configuring Route-Maps

Configuring Templates

About Route Profiles

A route profile specifies the route-control set actions used in import, export, and redistribute route-maps. Route profile templates can be defined either under the tenant or under the tenant VRF.

Configuring a Tenant-Scoped Route Profile

This procedure creates a tenant-scoped route profile that is used to configure BGP dampening and route redistribution.

Before you begin

- Configure a tenant and VRF.
- Enable VRF on a leaf.

SUMMARY STEPS

1. **configure**
2. **leaf *node-id***
3. **[no] template route-profile *profile-name* tenant *tenant-name***
4. **[no] set community {regular | extended} *value* {none | replace | additive}**
5. **[no] set dampening *half-life* *reuse* *suppress* *max-suppress-time***
6. **[no] set local-preference *value***

7. [no] set metric *value*
8. [no] set metric-type {*type-1* | *type-2*}
9. [no] set tag *name*
10. [no] set weight *weight*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	[no] template route-profile <i>profile-name</i> tenant <i>tenant-name</i> Example: <code>apic1(config-leaf)# template route-profile rp1 tenant exampleCorp</code>	Creates a tenant-scoped route profile.
Step 4	Required: [no] set community { regular extended } <i>value</i> { none replace additive } Example: <code>apic1(config-leaf-template-route-profile)# set community extended 20:22 additive</code>	Sets the BGP community attribute.
Step 5	Required: [no] set dampening <i>half-life</i> <i>reuse</i> <i>suppress</i> <i>max-suppress-time</i> Example: <code>apic1(config-leaf-template-route-profile)# set dampening 15 750 2000 60</code>	Configures route flap dampening behavior. The parameters are: • <i>half-life</i>—Decay half life, which is the time in minutes after which a penalty is decreased. Once the route has been assigned a penalty, the penalty is decreased by half after the half life period. The range is 1 to 60 minutes. • <i>reuse</i>—A route is unsuppressed (reused) if the penalty for a flapping route decreases enough to fall below this value. The range is 1 to 20000. • <i>suppress</i>—A route is suppressed when its penalty exceeds this limit. The range is 1 to 20000. • <i>max-suppress-time</i>—The maximum time in minutes that a stable route can be suppressed. The range is 1 to 255.
Step 6	Required: [no] set local-preference <i>value</i> Example:	Sets the BGP local preference value. The range is from 0 to 4294967295.

	Command or Action	Purpose
	<code>apic1(config-leaf-template-route-profile)# set local-preference 64</code>	
Step 7	Required: [no] set metric <i>value</i> Example: <code>apic1(config-leaf-template-route-profile)# set metric 128</code>	Sets the metric for the destination routing protocol.
Step 8	Required: [no] set metric-type {<i>type-1</i> <i>type-2</i>} Example: <code>apic1(config-leaf-template-route-profile)# set metric-type type-2</code>	The options are as follows: • type-1—OSPF external type 1 metric • type-2—OSPF external type 2 metric
Step 9	Required: [no] set tag <i>name</i> Example: <code>apic1(config-leaf-template-route-profile)# set tag 1111</code>	Sets the tag value for the destination routing protocol. The <i>name</i> parameter is an unsigned integer.
Step 10	Required: [no] set weight <i>weight</i> Example: <code>apic1(config-leaf-template-route-profile)# set weight 20</code>	Sets the tag value for the destination routing protocol. The <i>weight</i> parameter is an unsigned integer.

Examples

This example shows how to configure a tenant-scoped route profile.

```

apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# template route-profile rpl tenant exampleCorp
This template will be available on all leaves where tenant exampleCorp has a VRF deployment
apic1(config-leaf-template-route-profile)# set community extended 20:22 additive
apic1(config-leaf-template-route-profile)# set dampening 15 750 2000 60
apic1(config-leaf-template-route-profile)# set local-preference 64
apic1(config-leaf-template-route-profile)# set metric 128
apic1(config-leaf-template-route-profile)# set metric-type type-2
apic1(config-leaf-template-route-profile)# set tag 1111
apic1(config-leaf-template-route-profile)# set weight 20

```

Configuring a VRF-Scoped Route Profile

This procedure creates a VRF-scoped route profile including ‘default-export’ and ‘default-import’. This route profile can be attached to a bridge domain (BD) while ‘matching’ a bridge-domain inside a route map through the **inherit route-profile** command.



Note VRF-scoped route profiles name **default-export** and **default-import** values, which are automatically applied on the match statements on the respective export/import route-maps used in the same tenant VRF.

Before you begin

- Configure a tenant and VRF.
- Enable VRF on a leaf.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **[no] vrf context tenant** *tenant-name* **vrf** *vrf-name*
4. **[no] template route-profile** *profile-name*
5. **[no] set community** {**regular** | **extended**} {**no-advertise** | **no-export** | *value* {**none** | **replace** | **additive**}}
6. **[no] set local-preference** *value*
7. **[no] set metric** *value*
8. **[no] set metric-type** {**type-1** | **type-2**}
9. **[no] set tag** *name*
10. **[no] set weight** *weight*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	[no] vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-leaf)# vrf context tenant exampleCorp vrf vrf1</code>	Enables VRF on the leaf and enters VRF configuration mode.
Step 4	[no] template route-profile <i>profile-name</i> Example: <code>apic1(config-leaf-vrf)# template route-profile default-export</code>	Creates a VRF-scoped route profile.
Step 5	Required: [no] set community { regular extended } { no-advertise no-export <i>value</i> { none replace additive }}	Sets the BGP community attribute.

	Command or Action	Purpose
	Example: <pre>apic1(config-leaf-vrf-template-route-profile)# set community extended 20:22 additive</pre>	
Step 6	Required: [no] set local-preference value Example: <pre>apic1(config-tenant-vrf-route-profile)# set local-preference 64</pre>	Sets the BGP local preference value. The range is from 0 to 4294967295.
Step 7	Required: [no] set metric value Example: <pre>apic1(config-tenant-vrf-route-profile)# set metric 128</pre>	Sets the metric for the destination routing protocol.
Step 8	Required: [no] set metric-type {type-1 type-2} Example: <pre>apic1(config-tenant-vrf-route-profile)# set metric-type type-2</pre>	The options are as follows: • type-1—OSPF external type 1 metric • type-2—OSPF external type 2 metric
Step 9	Required: [no] set tag name Example: <pre>apic1(config-tenant-vrf-route-profile)# set tag 1111</pre>	Sets the tag value for the destination routing protocol. The <i>name</i> parameter is an unsigned integer.
Step 10	Required: [no] set weight weight Example: <pre>apic1(config-tenant-vrf-route-profile)# set weight 20</pre>	Sets the tag value for the destination routing protocol. The <i>weight</i> parameter is an unsigned integer.

Examples

This example shows how to configure a VRF-scoped route profile.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant exampleCorp vrf vrf1
apic1(config-leaf-vrf)# template route-profile default-export
apic1(config-leaf-vrf-template-route-profile)# set community extended 20:22 additive
apic1(config-leaf-vrf-template-route-profile)# set local-preference 64
apic1(config-leaf-vrf-template-route-profile)# set metric 128
apic1(config-leaf-vrf-template-route-profile)# set metric-type type-2
apic1(config-leaf-vrf-template-route-profile)# set tag 1111
apic1(config-leaf-vrf-template-route-profile)# set weight 20
```

Creating a Route-Map

Route-maps are created with a prefix-list on a per-tenant basis to indicate the bridge domain public subnets to be advertised to external routers. In addition, a prefix-list must be created to allow all transit routes to be advertised to an external router. The prefix-list for transit routes are configured by an administrator. The default behavior is to deny all transit route advertisement to an external router.

Before you begin

Configure a tenant and VRF.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **[no] vrf context tenant** *tenant-name* **vrf** *vrf-name*
4. (Optional) **[no] router-id** *ipv4-address*
5. **[no] route-map** *name*
6. **[no] ip prefix-list** *list-name* **permit** *prefix/masklen* [**le** {**32** | **128**}]
7. **[no] match prefix-list** *list-name*
8. **[no] set metric** *value*
9. **[no] set metric-type** {**type-1** | **type-2**}
10. **[no] set local-preference** *value*
11. **[no] set community** {**regular** | **extended**} *value* {**none** | **replace** | **additive**}
12. **[no] set tag** *name*
13. **[no] set weight** *value*
14. **[no] contract** {**provider** | **consumer**} *contract-name* [**imported**]
15. **[no] match route group** *group-name* [**order** *number*]
16. **[no] match bridge-domain** *bd-name*
17. **[no] inherit route-profile** *profile-name*
18. **[no] bridge-domain-match**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	[no] vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-leaf)# vrf context tenant exampleCorp vrf v1</code>	Configures a tenant VRF on the node.

	Command or Action	Purpose
Step 4	(Optional) [no] router-id <i>ipv4-address</i> Example: apic1(config-leaf-vrf) # router-id 1.2.3.4	Assigns a router ID for routing protocols running on the VRF. If you do not assign a router ID, an ID is generated internally that is unique to each leaf switch.
Step 5	Required: [no] route-map <i>name</i> Example: apic1(config-leaf-vrf) # route-map bgpMap	Creates a route-map and enters route-map configuration.
Step 6	Required: [no] ip prefix-list <i>list-name</i> permit <i>prefix/masklen</i> [le { 32 128 }] Example: apic1(config-leaf-vrf-route-map) # ip prefix-list list1 permit 13.13.13.0/24	Creates a prefix-list under the route-map.
Step 7	Required: [no] match prefix-list <i>list-name</i> Example: apic1(config-leaf-vrf-route-map) # match prefix-list list1	Matches a prefix-list that has already been created and enters the match mode to configure the route-control profile for the prefix-list.
Step 8	Required: [no] set metric <i>value</i> Example: apic1(config-leaf-vrf-route-map-match) # set metric 128	Sets the metric for the destination routing protocol.
Step 9	Required: [no] set metric-type { type-1 type-2 } Example: apic1(config-leaf-vrf-route-map-match) # set metric-type type-2	The options are as follows: • type-1—OSPF external type 1 metric • type-2—OSPF external type 2 metric
Step 10	Required: [no] set local-preference <i>value</i> Example: apic1(config-leaf-vrf-route-map-match) # set local-preference 64	Sets the BGP local preference value. The range is from 0 to 4294967295.
Step 11	Required: [no] set community { regular extended } <i>value</i> { none replace additive } Example: apic1(config-leaf-vrf-route-map-match) # set community extended 20:22 additive	Sets the community attribute for a BGP route update. Specify the <i>community-value</i> in <i>aa:nn</i> format. Specify the action as one of the following: • additive—Add to existing community • replace—Replace existing community • none—Do not change community
Step 12	Required: [no] set tag <i>name</i> Example:	Sets the tag value for the destination routing protocol. The <i>name</i> parameter is an unsigned integer.

	Command or Action	Purpose
	<code>apic1(config-leaf-vrf-route-map-match)# set tag 1111</code>	
Step 13	Required: [no] set weight value Example: <code>apic1(config-leaf-vrf-route-map-match)# set weight 32</code>	Specifies the BGP weight for the routing table.
Step 14	Required: [no] contract {provider consumer } contract-name [imported] Example: <code>apic1(config-leaf-vrf-route-map-match)# contract provider prov 1</code>	Add contract, required to leak routes (matching this prefix list) from the VRF.
Step 15	Required: [no] match route group group-name [order number] Example: <code>apic1(config-leaf-vrf-route-map)# match route group g1 order 1</code>	Matches a route group that has already been created and enters the match mode to configure the route-map. Repeat the steps 8-13 or only step 18 to configure the route map for the route group. See step 17 to inherit the route map instead of inline set actions.
Step 16	Required: [no] match bridge-domain bd-name Example: <code>apic1(config-leaf-vrf-route-map)# bridge-domain bd1</code>	Matches a bridge domain in order to export its public subnets through the protocol.
Step 17	Required: [no] inherit route-profile profile-name Example: <code>apic1(config-leaf-vrf-route-map-match)# inherit route-profile rp1</code>	Configures route map for bridge domain. Note The route map was already created using the command template route-profile .
Step 18	Required: [no] bridge-domain-match Example: <code>apic1(config-leaf-vrf-route-map)# no bridge-domain-match</code>	Configures route map for bridge domain. Note Disables the bridge domain (BD) match in a route map, eliminating the need to delete the BD configuration from the route map. This is required if there are BDs matched in a route map, and the route map is used to filter out the BD subnets using route group/explicit prefix list.

Examples

This example shows how to create a route-map and add/match a prefix-list, a community-list, and a bridge-domain.

```
# CREATE A ROUTE-MAP
apic1# configure
apic1(config)# leaf 101
```

```

apicl(config-leaf)# vrf context tenant exampleCorp vrf v1
apicl(config-leaf-vrf)# route-map bgpMap

# CREATE A PREFIX-LIST
apicl(config-leaf-vrf-route-map)# ip prefix-list list1 permit 13.13.13.0/24
apicl(config-leaf-vrf-route-map)# ip prefix-list list1 permit 14.14.14.0/24

# MATCH THE PREFIX-LIST
apicl(config-leaf-vrf-route-map)# match prefix-list list1

# CONFIGURE A ROUTE-PROFILE FOR THE PREFIX-LIST
apicl(config-leaf-vrf-route-map-match)# set metric 128
apicl(config-leaf-vrf-route-map-match)# set metric-type type-2
apicl(config-leaf-vrf-route-map-match)# set local-preference 64
apicl(config-leaf-vrf-route-map-match)# set community extended 20:22 additive
apicl(config-leaf-vrf-route-map-match)# set tag 1111
apicl(config-leaf-vrf-route-map-match)# set weight 32
apicl(config-leaf-vrf-route-map-match)# contract provider prov 1

# CREATE COMMUNITY LIST
apicl# configure
apicl(config)# leaf 101
apicl(config-leaf)# template community-list standard CL_1 65536:20 tenant exampleCorp

# CREATE ROUTE GROUP
apicl# configure
apicl(config)# leaf 101
apicl(config-leaf)# template route group g1 tenant exampleCorp
apicl(config-route-group)# ip prefix permit 15.15.15.0/24
apicl(config-route-group)# community-list standard 65535:20

# MATCH ROUTE GROUP
apicl(config-leaf)# vrf context tenant exampleCorp vrf v1
apicl(config-leaf-vrf)# route-map bgpMap
apicl(config-leaf-vrf-route-map)# match route group g1 order 1

# CONFIGURE ROUTE PROFILE FOR COMMUNITY-LIST
apicl(config-leaf-vrf-route-map-match)# set metric 128
apicl(config-leaf-vrf-route-map-match)# set metric-type type-2
apicl(config-leaf-vrf-route-map-match)# set local-preference 64
apicl(config-leaf-vrf-route-map-match)# set community extended 20:22 additive
apicl(config-leaf-vrf-route-map-match)# set tag 1111
apicl(config-leaf-vrf-route-map-match)# set weight 32

# CONFIGURE ROUTE PROFILE ROUTE GROUP
apicl(config-leaf-vrf-route-map-match)# set metric 128
apicl(config-leaf-vrf-route-map-match)# set metric-type type-2
apicl(config-leaf-vrf-route-map-match)# set local-preference 64
apicl(config-leaf-vrf-route-map-match)# set community extended 20:22 additive
apicl(config-leaf-vrf-route-map-match)# set tag 1111
apicl(config-leaf-vrf-route-map-match)# set weight 32

# Or CREATE A ROUTE PROFILE TEMPLATE AND INHERIT IT FOR ROUTE GROUP
apicl(config)# leaf 101
apicl(config-leaf)# vrf context tenant exampleCorp vrf v1
apicl(config-leaf-vrf)# template route-profile rp1
apicl(config-leaf-vrf-template-route-profile)# set metric 128
apicl(config-leaf-vrf-template-route-profile)# set metric-type type-2
apicl(config-leaf-vrf-template-route-profile)# set local-preference 64
apicl(config-leaf-vrf-template-route-profile)# set community extended 20:22 additive

```

```

apic1(config-leaf-vrf-template-route-profile)# set tag 1111
apic1(config-leaf-vrf-template-route-profile)# set weight 32
apic1(config-leaf-vrf-template-route-profile)# exit

apic1(config-leaf-vrf)# route-map bgpMap
apic1(config-leaf-vrf-route-map)# match route group g1 order 1
apic1(config-leaf-vrf-route-map-match)# inherit route-profile rp1

# CREATE A BRIDGE-DOMAIN
apic1# configure
apic1(config)# tenant exampleCorp
apic1(config-tenant)# vrf context v1
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# bridge-domain bd1
apic1(config-tenant-bd)# vrf member v1
apic1(config-tenant-bd)# exit
apic1(config-tenant)# interface bridge-domain bd1
apic1(config-tenant-interface)# ip address 13.13.13.1/24 scope public
apic1(config-tenant-interface)# exit
apic1(config-tenant)# exit

# CREATE A ROUTE-PROFILE FOR THE BRIDGE-DOMAIN
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant exampleCorp vrf v1
apic1(config-leaf-vrf)# template route-profile default-export
apic1(config-leaf-vrf-template-route-profile)# set metric 128
apic1(config-leaf-vrf-template-route-profile)# set metric-type type-2
apic1(config-leaf-vrf-template-route-profile)# set local-preference 64
apic1(config-leaf-vrf-template-route-profile)# set community extended 20:22 additive
apic1(config-leaf-vrf-template-route-profile)# set tag 1111
apic1(config-leaf-vrf-template-route-profile)# set weight 20
apic1(config-leaf-vrf-template-route-profile)# exit

# MATCH THE BRIDGE-DOMAIN
apic1(config-leaf-vrf)# route-map bgpMap
apic1(config-leaf-vrf-route-map)# match bridge-domain bd1

# CONFIGURE A ROUTE-PROFILE FOR THE BRIDGE-DOMAIN
apic1(config-leaf-vrf-route-map-match)# inherit route-profile default-export

```

Configuring Route-Maps in Routing Protocols

The OSPF, BGP, and EIGRP routing protocols use route-maps to filter routes for import and export. For the general steps required to configure these protocols, see the documentation sections for each. To configure route-maps in these protocols, use the following commands and see the examples.

Protocol	Route-Map Command
BGP	[no] route-map <i>map-name</i> { in out }
OSPF	[no] area <i>area-id</i> route-map <i>map-name</i> { in out }
EIGRP	[no] ip distribute list default route-map <i>map-name</i> out

Examples

This example shows how to configure a route-map in BGP, OSPF and EIGRP.

```
# BGP
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# router bgp 100
apic1(config-bgp)# vrf member tenant exampleCorp vrf v1
apic1(config-leaf-bgp-vrf)# neighbor 3.3.3.3
apic1(config-leaf-bgp-vrf-neighbor)# route-map map1 out
apic1(config-leaf-bgp-vrf-neighbor)# route-map map2 in
apic1(config-leaf-bgp-vrf-neighbor)# exit
apic1(config-leaf-bgp-vrf)# exit
apic1(config-bgp)# exit
apic1(config-leaf)# exit

# OSPF
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant exampleCorp vrf v1
apic1(config-leaf-ospf-vrf)# area 0.0.0.1 route-map map1 out
apic1(config-leaf-ospf-vrf)# area 0.0.0.1 route-map map2 in
apic1(config-leaf-ospf-vrf)# exit
apic1(config-leaf-ospf)# exit
apic1(config-leaf)# exit

#EIGRP
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/3
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member tenant exampleCorp vrf v1
apic1(config-leaf-if)# ip address 13.13.13.13/24
apic1(config-leaf-if)# ip router eigrp default
apic1(config-leaf-if)# ip distribute-list eigrp default route-map map1 out
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

Configuring an Export Map (Inter-VRF Route Leak)

Before you begin

- Create a route-map.
- Add prefix-list(s) to the route-map containing prefixes matching routes that need to be leaked.
- Match the prefix-list(s) and add the contract(s) to enable the route leak.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*

3. **[no] vrf context tenant** *tenant-name* **vrf** *vrf-name*
4. **[no] export map** *map-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	[no] vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-leaf)# vrf context tenant exampleCorp vrf v1</code>	Configures a tenant VRF on the node.
Step 4	[no] export map <i>map-name</i> Example: <code>apic1(config-leaf-vrf)# export map shared-route-map1</code>	Configures route-map in this VRF to export (leak) routes from this VRF into consumer VRFs.

Examples

This example shows how to create and export a route-map.

```
# CREATE A ROUTE-MAP
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant exampleCorp vrf v1
apic1(config-leaf-vrf)# router-id 1.2.3.4
apic1(config-leaf-vrf)# route-map shared-route-map1
apic1(config-leaf-vrf-route-map)# ip prefix-list list1 permit 13.13.13.0/24
apic1(config-leaf-vrf-route-map)# match prefix-list list1
apic1(config-leaf-vrf-route-map-match)# contract provider prov1
apic1(config-leaf-vrf-route-map-match)# exit
apic1(config-leaf-vrf-route-map)# exit
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# exit

# EXPORT THE ROUTE-MAP
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant exampleCorp vrf v1
apic1(config-leaf-vrf)# export map shared-route-map1
```

Configuring Bi-Directional Route Forwarding (BFD)

About BFD

Bidirectional Forwarding Detection (BFD) is a detection protocol designed to provide fast forwarding-path failure detection times for media types, encapsulations, topologies, and routing protocols. You can use BFD to detect forwarding path failures at a uniform rate, rather than the variable rates for different protocol hello mechanisms. BFD makes network profiling and planning easier and reconvergence time consistent and predictable.

Use Bidirectional Forwarding Detection (BFD) to provide sub-second failure detection times in the forwarding path between ACI fabric border leaf switches configured to support peering router connections.

Configuring BFD Globally

You can configure the BFD session parameters for all BFD sessions on the device. The BFD session parameters are negotiated between the BFD peers in a three-way handshake.

To configure BFD globally, perform the following procedures:

- Configure the BFD global configuration settings
- Configure an access leaf policy group and inherit the previously created BFD global policies
- Associate the previously created leaf policy group onto a leaf switch or group of leaf switches

SUMMARY STEPS

1. **configure**
2. **[no] template bfd {ip | ipv6} *global-policy-name***
3. **[no] echo-address *ip-address***
4. **[no] slow-timer *milliseconds***
5. **[no] min-tx *milliseconds***
6. **[no] min-rx *milliseconds***
7. **[no] multiplier *policy-name***
8. **[no] echo-rx-interval *policy-name***
9. **exit**
10. **[no] template leaf-policy-group *leaf-policy-name***
11. **[no] inherit bfd {ip | ipv6} *global-policy-name***
12. **exit**
13. **[no] leaf-profile *leaf-profile-name***
14. **[no] leaf-group *leaf-group-name***
15. **[no] leaf-policy-group *leaf-policy-name***
16. **[no] leaf *leaf-range***

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	[no] template bfd {ip ipv6} global-policy-name Example: <code>apic1(config)# template bfd ip bfd_global</code>	Creates a BFD policy template.
Step 3	[no] echo-address ip-address Example: <code>apic1(config-bfd)# echo-address 192.0.20.123</code> <code>apic1(config-bfd)# echo-address 34::1/64</code>	Specifies the IP address to use as the source address for BFD echo packets.
Step 4	[no] slow-timer milliseconds Example: <code>apic1(config-bfd)# slow-timer 2000</code>	Configures the slow timer used in the echo function. This value determines how fast BFD starts up a new sessions and at what speed the asynchronous sessions use for BFD control packets when the echo function is enabled. The slow-timer value is used as the new control packet interval, while the echo packets use the configured BFD intervals. The echo packets are used for link failure detection, while the control packets at the slower rate maintain the BFD session. The range is from 1000 to 30000 milliseconds.
Step 5	[no] min-tx milliseconds Example: <code>apic1(config-bfd)# min-tx 100</code>	Specifies the interval at which this device wants to send BFD hello messages. The range is 50 to 999 milliseconds.
Step 6	[no] min-rx milliseconds Example: <code>apic1(config-bfd)# min-rx 70</code>	Specifies the minimum interval at which this device can accept BFD hello messages from another BFD device. The range is 50 to 999 milliseconds.
Step 7	[no] multiplier policy-name Example: <code>apic1(config-bfd)# multiplier 3</code>	Specifies the number of missing BFD hello messages from another BFD device before this local device detects a fault in the forwarding path. The range is 1 to 50.
Step 8	[no] echo-rx-interval policy-name Example: <code>apic1(config-bfd)# echo-rx-interval 500</code>	Specifies the minimum interval between received BFD echo packets that this system is capable of supporting. The range is 50 to 999 milliseconds.
Step 9	exit Example: <code>apic1(config-bfd)# exit</code>	Returns to global configuration mode.

	Command or Action	Purpose
Step 10	[no] template leaf-policy-group <i>leaf-policy-name</i> Example: apic1(config)# template leaf-policy-group leaf_pg1	Configures an access leaf policy group.
Step 11	[no] inherit bfd {ip ipv6} <i>global-policy-name</i> Example: apic1(config-leaf-policy-group)# inherit bfd ip bfd_global	Inherits the previously created BFD global policies.
Step 12	exit Example: apic1(config-leaf-policy-group)# exit	Returns to global configuration mode.
Step 13	[no] leaf-profile <i>leaf-profile-name</i> Example: apic1(config)# leaf-profile leaf_profile1	Configures a leaf profile.
Step 14	[no] leaf-group <i>leaf-group-name</i> Example: apic1(config-leaf-profile)# leaf-group leaf_group1	Creates or specifies a group of leaf switches.
Step 15	[no] leaf-policy-group <i>leaf-policy-name</i> Example: apic1(config-leaf-group)# leaf-policy-group leaf_pg1	Specifies the previously created leaf policy group to be associated to the leaf switches.
Step 16	[no] leaf <i>leaf-range</i> Example: apic1(config-leaf-group)# leaf 101-102	Adds one or more leaf switches to the leaf switch group.

Examples

This example shows how to configure BFD globally and apply it to a group of leaf switches.

```
# CONFIGURE BFD GLOBAL POLICIES
apic1# configure
apic1(config)# template bfd ip bfd_global
apic1(config-bfd)# echo-address 192.0.20.123
apic1(config-bfd)# slow-timer 2000
apic1(config-bfd)# min-tx 100
apic1(config-bfd)# min-rx 70
apic1(config-bfd)# multiplier 3
apic1(config-bfd)# echo-rx-interval 500
apic1(config-bfd)# exit

# CONFIGURE AN ACCESS LEAF POLICY GROUP AND INHERIT BFD GLOBAL POLICIES
apic1(config)# template leaf-policy-group leaf_pg1
```



```

apicl(config-leaf-policy-group)# inherit bfd ip bfd_global
apicl(config-leaf-policy-group)# exit

# CONFIGURE A LEAF GROUP AND ASSOCIATE THE LEAF POLICY GROUP
apicl(config)# leaf-profile leaf_profile1
apicl(config-leaf-profile)# leaf-group leaf_group1
apicl(config-leaf-group)# leaf-policy-group leaf_pg1
apicl(config-leaf-group)# leaf 101-102

```

Configuring BFD Globally on Leaf Switch Using the NX-OS Style CLI

Step 1 To configure the BFD IPV4 global configuration (bfdIpv4InstPol) using the NX-OS CLI:

Example:

```

apicl# configure
apicl(config)# template bfd ip bfd_ipv4_global_policy
apicl(config-bfd)# [no] echo-address 1.2.3.4
apicl(config-bfd)# [no] slow-timer 2500
apicl(config-bfd)# [no] min-tx 100
apicl(config-bfd)# [no] min-rx 70
apicl(config-bfd)# [no] multiplier 3
apicl(config-bfd)# [no] echo-rx-interval 500
apicl(config-bfd)# exit

```

Step 2 To configure the BFD IPV6 global configuration (bfdIpv6InstPol) using the NX-OS CLI:

Example:

```

apicl# configure
apicl(config)# template bfd ipv6 bfd_ipv6_global_policy
apicl(config-bfd)# [no] echo-address 34::1/64
apicl(config-bfd)# [no] slow-timer 2500
apicl(config-bfd)# [no] min-tx 100
apicl(config-bfd)# [no] min-rx 70
apicl(config-bfd)# [no] multiplier 3
apicl(config-bfd)# [no] echo-rx-interval 500
apicl(config-bfd)# exit

```

Step 3 To configure access leaf policy group (infraAccNodePGrp) and inherit the previously created BFD global policies using the NX-OS CLI:

Example:

```

apicl# configure
apicl(config)# template leaf-policy-group test_leaf_policy_group
apicl(config-leaf-policy-group)# [no] inherit bfd ip bfd_ipv4_global_policy
apicl(config-leaf-policy-group)# [no] inherit bfd ipv6 bfd_ipv6_global_policy
apicl(config-leaf-policy-group)# exit

```

Step 4 To associate the previously created leaf policy group onto a leaf using the NX-OS CLI:

Example:

```

apicl(config)# leaf-profile test_leaf_profile
apicl(config-leaf-profile)# leaf-group test_leaf_group
apicl(config-leaf-group)# leaf-policy-group test_leaf_policy_group

```

```
apic1(config-leaf-group)# leaf 101-102
apic1(config-leaf-group)# exit
```

Configuring BFD Globally on Spine Switch Using the NX-OS Style CLI

Use this procedure to configure BFD globally on spine switch using the NX-OS style CLI.

Step 1 To configure the BFD IPV4 global configuration (bfdIpv4InstPol) using the NX-OS CLI:

Example:

```
apic1# configure
apic1(config)# template bfd ip bfd_ipv4_global_policy
apic1(config-bfd)# [no] echo-address 1.2.3.4
apic1(config-bfd)# [no] slow-timer 2500
apic1(config-bfd)# [no] min-tx 100
apic1(config-bfd)# [no] min-rx 70
apic1(config-bfd)# [no] multiplier 3
apic1(config-bfd)# [no] echo-rx-interval 500
apic1(config-bfd)# exit
```

Step 2 To configure the BFD IPV6 global configuration (bfdIpv6InstPol) using the NX-OS CLI:

Example:

```
apic1# configure
apic1(config)# template bfd ipv6 bfd_ipv6_global_policy
apic1(config-bfd)# [no] echo-address 34::1/64
apic1(config-bfd)# [no] slow-timer 2500
apic1(config-bfd)# [no] min-tx 100
apic1(config-bfd)# [no] min-rx 70
apic1(config-bfd)# [no] multiplier 3
apic1(config-bfd)# [no] echo-rx-interval 500
apic1(config-bfd)# exit
```

Step 3 To configure spine policy group and inherit the previously created BFD global policies using the NX-OS CLI:

Example:

```
apic1# configure
apic1(config)# template spine-policy-group test_spine_policy_group
apic1(config-spine-policy-group)# [no] inherit bfd ip bfd_ipv4_global_policy
apic1(config-spine-policy-group)# [no] inherit bfd ipv6 bfd_ipv6_global_policy
apic1(config-spine-policy-group)# exit
```

Step 4 To associate the previously created spine policy group onto a spine switch using the NX-OS CLI:

Example:

```
apic1# configure
apic1(config)# spine-profile test_spine_profile
apic1(config-spine-profile)# spine-group test_spine_group
apic1(config-spine-group)# spine-policy-group test_spine_policy_group
apic1(config-spine-group)# spine 103-104
apic1(config-leaf-group)# exit
```

Overriding Global BFD Settings

Configuring BFD Interface Override Policy

There are three supported interfaces (routed L3 interfaces, the external SVI interface, and the routed sub-interfaces) on which you can configure an explicit BFD configuration. If you don't want to use the global configuration, yet you want to have an explicit configuration on a given interface, you can create your own global configuration, which gets applied to all the interfaces on a specific switch or set of switches. This interface override configuration should be used if you want more granularity on a specific switch on a specific interface.

Before you begin

A tenant has already been created.

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **vrf context** *vrf-name*
4. **exit**
5. **exit**
6. **leaf** *node-id*
7. **[no] vrf context tenant** *tenant-name* **vrf** *vrf-name*
8. **exit**
9. **[no] interface** *type*
10. **[no] vrf member tenant** *tenant-name* **vrf** *vrf-name*
11. **exit**
12. **[no] template bfd** *template-name* **tenant** *tenant-name*
13. **[no] echo-mode** **enable**
14. **[no] echo-rx-interval** *policy-name*
15. **[no] min-tx** *milliseconds*
16. **[no] min-rx** *milliseconds*
17. **[no] multiplier** *policy-name*
18. **[no] optimize subinterface**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apic1(config)# tenant exampleCorp</code>	Specifies the tenant to be configured.

	Command or Action	Purpose
Step 3	vrf context <i>vrf-name</i> Example: <code>apic1(config-tenant) # vrf context vrf1</code>	Associates a VRF with the tenant.
Step 4	exit Example: <code>apic1(config-tenant-vrf) # exit</code>	Returns to tenant configuration mode.
Step 5	exit Example: <code>apic1(config-tenant) # exit</code>	Returns to global configuration mode.
Step 6	leaf <i>node-id</i> Example: <code>apic1(config) # leaf 101</code>	Specifies the leaf to be configured.
Step 7	[no] vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-leaf) # vrf context tenant exampleCorp vrf vrf1</code>	Configures a tenant VRF on the node.
Step 8	exit Example: <code>apic1(config-leaf-vrf) # exit</code>	Returns to leaf configuration mode.
Step 9	[no] interface <i>type</i> Example: <code>apic1(config-leaf) # interface eth 1/18</code>	Enters interface configuration mode.
Step 10	[no] vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-leaf-if) # vrf member tenant exampleCorp vrf vrf1</code>	
Step 11	exit Example: <code>apic1(config-leaf-if) # exit</code>	Returns to leaf configuration mode.
Step 12	[no] template bfd <i>template-name</i> tenant <i>tenant-name</i> Example: <code>apic1(config-leaf) # template bfd bfdIfPoll tenant exampleCorp</code>	Configures a BFD interface policy.
Step 13	[no] echo-mode enable Example:	Enables or disables the sending of BFD echo packets in addition to BFD control packets.

	Command or Action	Purpose
	<code>apic1(config-template-bfd-pol) # echo-mode enable</code>	
Step 14	[no] echo-rx-interval <i>policy-name</i> Example: <pre>apic1(config-template-bfd-pol) # echo-rx-interval 500</pre>	Specifies the minimum interval between received BFD echo packets that this system is capable of supporting. The range is 50 to 999 milliseconds.
Step 15	[no] min-tx <i>milliseconds</i> Example: <pre>apic1(config-template-bfd-pol) # min-tx 100</pre>	Specifies the interval at which this device sends BFD hello messages. The range is 50 to 999 milliseconds.
Step 16	[no] min-rx <i>milliseconds</i> Example: <pre>apic1(config-template-bfd-pol) # min-rx 70</pre>	Specifies the minimum interval at which this device can accept BFD hello messages from another BFD device. The range is 50 to 999 milliseconds.
Step 17	[no] multiplier <i>policy-name</i> Example: <pre>apic1(config-template-bfd-pol) # multiplier 5</pre>	Specifies the number of missing BFD hello messages from another BFD device before this local device detects a fault in the forwarding path. The range is 1 to 50.
Step 18	[no] optimize subinterface Example: <pre>apic1(config-template-bfd-pol) # optimize subinterface</pre>	Enables or disables sub-interface optimization. BFD creates sessions for all configured subinterfaces. BFD sets the subinterface with the lowest configured VLAN ID as the master subinterface and that subinterface uses the BFD session parameters of the parent interface. The remaining subinterfaces use the slow timer. If the optimized subinterface session detects an error, BFD marks all subinterfaces on that physical interface as down.

Examples

This example shows how to configure a BFD override policy and apply it to an interface.

```
apic1# configure
apic1(config)# tenant exampleCorp
apic1(config-tenant)# vrf context vrf1
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant exampleCorp vrf vrf1
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# interface eth 1/18
apic1(config-leaf-if)# vrf member tenant exampleCorp vrf vrf1
apic1(config-leaf-if)# exit

# CONFIGURE BFD INTERFACE OVERRIDE POLICY
apic1(config-leaf)# template bfd bfdIfPol1 tenant exampleCorp
apic1(config-template-bfd-pol) # echo-mode enable
apic1(config-template-bfd-pol) # echo-rx-interval 500
apic1(config-template-bfd-pol) # min-tx 100
apic1(config-template-bfd-pol) # min-rx 70
```

```
apicl(config-template-bfd-pol)# multiplier 5
apicl(config-template-bfd-pol)# optimize subinterface
```

Applying the BFD Interface Override Policy to Interfaces

You can apply a BFD interface override policy to routed L3 interfaces, the external SVI interface, and the routed sub-interfaces.

Before you begin

A BFD interface override policy has already been created.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **[no] interface** *type*
4. **[no] ipv6 address** *ipv6-address* **[preferred]**
5. **[no] vrf member tenant** *tenant-name* **vrf** *vrf-name*
6. **bfd {ip | ipv6} tenant mode**
7. **bfd {ip | ipv6} inherit interface-policy** *policy-name*
8. **bfd {ip | ipv6} authentication keyed-sha1** *keyid* *keyid* **key** *key*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: apicl(config)# leaf 101	Specifies the leaf to be configured.
Step 3	[no] interface <i>type</i> Example: apicl(config-leaf)# interface Ethernet 1/15	Enters interface configuration mode. Supported interfaces are routed L3 interfaces, the external SVI interface, and the routed sub-interfaces.
Step 4	[no] ipv6 address <i>ipv6-address</i> [preferred] Example: apicl(config-leaf-if)# ipv6 address 2001::10:1/64 preferred	Specifies an IP address to be the default source address for traffic from the interface. Note This command is used only if the interface is an IPv6 interface.
Step 5	[no] vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: apicl(config-leaf-if)# vrf member tenant exampleCorp vrf vrf1	Attaches the interface to the tenant VRF. Note This command is used only if the interface is a VLAN interface.

	Command or Action	Purpose
Step 6	bfd {ip ipv6} tenant mode Example: <code>apic1(config-leaf-if)# bfd ip tenant mode</code>	Enables BFD tenant mode.
Step 7	bfd {ip ipv6} inherit interface-policy <i>policy-name</i> Example: <code>apic1(config-leaf-if)# bfd ip inherit interface-policy bfdIfPoll</code>	Inherits the specified BFD interface template policy.
Step 8	bfd {ip ipv6} authentication keyed-sha1 keyid <i>keyid</i> key <i>key</i> Example: <code>apic1(config-leaf-if)# bfd ip authentication keyed-sha1 key 10 key password</code>	Configures BFD authentication as keyed SHA-1.

Examples

This example shows how to inherit the previously created BFD interface policy onto a L3 interface with an IPv4 address.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface eth 1/15
apic1(config-leaf-if)# bfd ip tenant mode
apic1(config-leaf-if)# bfd ip inherit interface-policy bfdIfPoll
apic1(config-leaf-if)# bfd ip authentication keyed-sha1 key 10 key password
```

This example shows how to inherit the previously created BFD interface policy onto a L3 interface with an IPv6 address.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface eth 1/15
apic1(config-leaf-if)# ipv6 address 2001::10:1/64 preferred
apic1(config-leaf-if)# bfd ip tenant mode
apic1(config-leaf-if)# bfd ip inherit interface-policy bfdIfPoll
apic1(config-leaf-if)# bfd ip authentication keyed-sha1 key 10 key password
```

This example shows how to configure BFD on a VLAN interface with an IPv4 address.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface vlan 15
apic1(config-leaf-if)# vrf member tenant exampleCorp vrf vrf1
apic1(config-leaf-if)# bfd ip tenant mode
apic1(config-leaf-if)# bfd ip inherit interface-policy bfdIfPoll
apic1(config-leaf-if)# bfd ip authentication keyed-sha1 key 10 key password
```

This example shows how to configure BFD on a VLAN interface with an IPv6 address.

```

apicl# configure
apicl(config)# leaf 101
apicl(config-leaf)# interface vlan 15
apicl(config-leaf-if)# ipv6 address 2001::10:1/64 preferred
apicl(config-leaf-if)# vrf member tenant exampleCorp vrf vrf1
apicl(config-leaf-if)# bfd ip tenant mode
apicl(config-leaf-if)# bfd ip inherit interface-policy bfdIfPol1
apicl(config-leaf-if)# bfd ip authentication keyed-sha1 key 10 key password

```

Enabling BFD on Consumer Protocols

These procedures provide the steps to enable BFD in the four consumer protocols (BGP, EIGRP, OSPF, and Static Routes), which are consumers of the BFD feature.

Enabling BFD on the BGP Consumer Protocol

Before you begin

A tenant has already been created.

SUMMARY STEPS

1. **configure**
2. **bgp-fabric**
3. **asn** *asn-number*
4. **exit**
5. **leaf** *node-id*
6. **router bgp** *asn-number*
7. **vrf member tenant** *tenant-name* **vrf** *vrf-name*
8. **neighbor** *ip-address*[/*masklength*]
9. **[no] bfd enable**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	bgp-fabric Example: apicl(config-bgp-fabric)# bgp-fabric	Enters BGP configuration mode for the fabric.
Step 3	asn <i>asn-number</i> Example: apicl(config-bgp-fabric)# asn 200	Specifies the BGP autonomous system number (ASN).

	Command or Action	Purpose
Step 4	exit Example: <code>apic1(config-bgp-fabric) # exit</code>	Returns to global configuration mode.
Step 5	leaf <i>node-id</i> Example: <code>apic1(config) # leaf 101</code>	Specifies the leaf to be configured.
Step 6	router bgp <i>asn-number</i> Example: <code>apic1(config-leaf) # router bgp 200</code>	Enters BGP policy configuration.
Step 7	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-bgp) # vrf member tenant exampleCorp vrf v100</code>	Specifies the VRF instance to associate with subsequent policy configuration mode commands.
Step 8	neighbor <i>ip-address</i>[/<i>masklength</i>] Example: <code>apic1(config-leaf-bgp-vrf) # neighbor 1.2.3.4</code>	Specifies the IP address of the neighbor. The mask length must be 32.
Step 9	[no] bfd enable Example: <code>apic1(config-leaf-bgp-vrf-neighbor) # bfd enable</code>	Enables or disables BFD on the BGP consumer protocol.

Examples

This example shows how to enable BFD on the BGP consumer protocol.

```
apic1# configure
apic1(config) # bgp-fabric
apic1(config-bgp-fabric) # asn 200
apic1(config-bgp-fabric) # exit
apic1(config) # leaf 101
apic1(config-leaf) # router bgp 200
apic1(config-bgp) # vrf member tenant exampleCorp vrf v100
apic1(config-leaf-bgp-vrf) # neighbor 1.2.3.4
apic1(config-leaf-bgp-vrf-neighbor) # bfd enable
```

Enabling BFD on the EIGRP Consumer Protocol

SUMMARY STEPS

1. **configure**
2. **leaf *node-id***

3. [no] interface *type*
4. [no] {ip | ipv6} bfd eigrp enable

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: apicl(config)# leaf 101	Specifies the leaf to be configured.
Step 3	[no] interface <i>type</i> Example: apicl(config-leaf)# interface Ethernet 1/15	Enters interface configuration mode.
Step 4	[no] {ip ipv6} bfd eigrp enable Example: apicl(config-leaf-if)# ip bfd eigrp enable	Enables or disables BFD on the EIGRP consumer protocol.

Examples

This example shows how to enable BFD on the EIGRP consumer protocol.

```

apicl# configure
apicl(config)# leaf 101
apicl(config-leaf)# interface eth 1/15
apicl(config-leaf-if)# ip bfd eigrp enable

```

Enabling BFD on the OSPF Consumer Protocol

SUMMARY STEPS

1. configure
2. leaf *node-id*
3. [no] interface *type*
4. [no] ip ospf bfd enable

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example:	Enters configuration mode.

	Command or Action	Purpose
	<code>apic1# configure</code>	
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	[no] interface <i>type</i> Example: <code>apic1(config-leaf)# interface vlan 123</code>	Enters interface configuration mode.
Step 4	[no] ip ospf bfd enable Example: <code>apic1(config-leaf-if)# ip ospf bfd enable</code>	Enables or disables BFD on the OSPF consumer protocol.

Examples

This example shows how to enable BFD on the OSPF consumer protocol.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface vlan 123
apic1(config-leaf-if)# ip ospf bfd enable
```

Enabling BFD on the Static Route Consumer Protocol

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **[no] vrf context tenant** *tenant-name* **vrf** *vrf-name*
4. **[no] {ip | ipv6} route** *ip-prefix/masklen* *next-hop-address* **bfd**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.

	Command or Action	Purpose
Step 3	[no] vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <pre>apic1(config-leaf)# vrf context tenant exampleCorp vrf vrf1</pre>	Configures a tenant VRF on the node.
Step 4	[no] {ip ipv6} route <i>ip-prefix/masklen next-hop-address</i> bfd Example: <pre>apic1(config-leaf-vrf)# ip route 10.0.0.1/16 10.0.0.5 bfd</pre>	Enables or disables BFD on the static route consumer protocol.

Examples

This example shows how to enable BFD on the static route consumer protocol.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant exampleCorp vrf vrf1
apic1(config-leaf-vrf)# ip route 10.0.0.1/16 10.0.0.5 bfd
```

Configuring BFD Consumer Protocols Using the NX-OS Style CLI

Step 1 To enable BFD on the BGP consumer protocol using the NX-OS CLI:

Example:

```
apic1# configure
apic1(config)# bgp-fabric
apic1(config-bgp-fabric)# asn 200
apic1(config-bgp-fabric)# exit
apic1(config)# leaf 101
apic1(config-leaf)# router bgp 200
apic1(config-bgp)# vrf member tenant t0 vrf v0
apic1(config-leaf-bgp-vrf)# neighbor 1.2.3.4
apic1(config-leaf-bgp-vrf-neighbor)# [no] bfd enable
```

Step 2 To enable BFD on the EIGRP consumer protocol using the NX-OS CLI:

Example:

```
apic1(config-leaf-if)# [no] ip bfd eigrp enable
```

Step 3 To enable BFD on the OSPF consumer protocol using the NX-OS CLI:

Example:

```
apic1(config-leaf-if)# [no] ip ospf bfd enable
```

```
apicl# configure
apicl(config)# spine 103
apicl(config-spine)# interface ethernet 5/3.4
apicl(config-spine-if) # [no] ip ospf bfd enable
```

Step 4 To enable BFD on the Static Route consumer protocol using the NX-OS CLI:

Example:

```
apicl(config-leaf-vrf)# [no] ip route 10.0.0.1/16 10.0.0.5 bfd

apicl(config)# spine 103
apicl(config-spine)# vrf context tenant infra vrf overlay-1
apicl(config-spine-vrf)# [no] ip route 21.1.1.1/32 32.1.1.1 bfd
```

Step 5 To enable BFD on IS-IS consumer protocol using the NX-OS CLI:

Example:

```
apicl(config)# leaf 101
apicl(config-spine)# interface ethernet 1/49
apicl(config-spine-if) # isis bfd enabled
apicl(config-spine-if) # exit
apicl(config-spine) # exit

apicl(config)# spine 103
apicl(config-spine)# interface ethernet 5/2
apicl(config-spine-if) # isis bfd enabled
apicl(config-spine-if) # exit
apicl(config-spine) # exit
```

Configuring Layer 3 Multicast

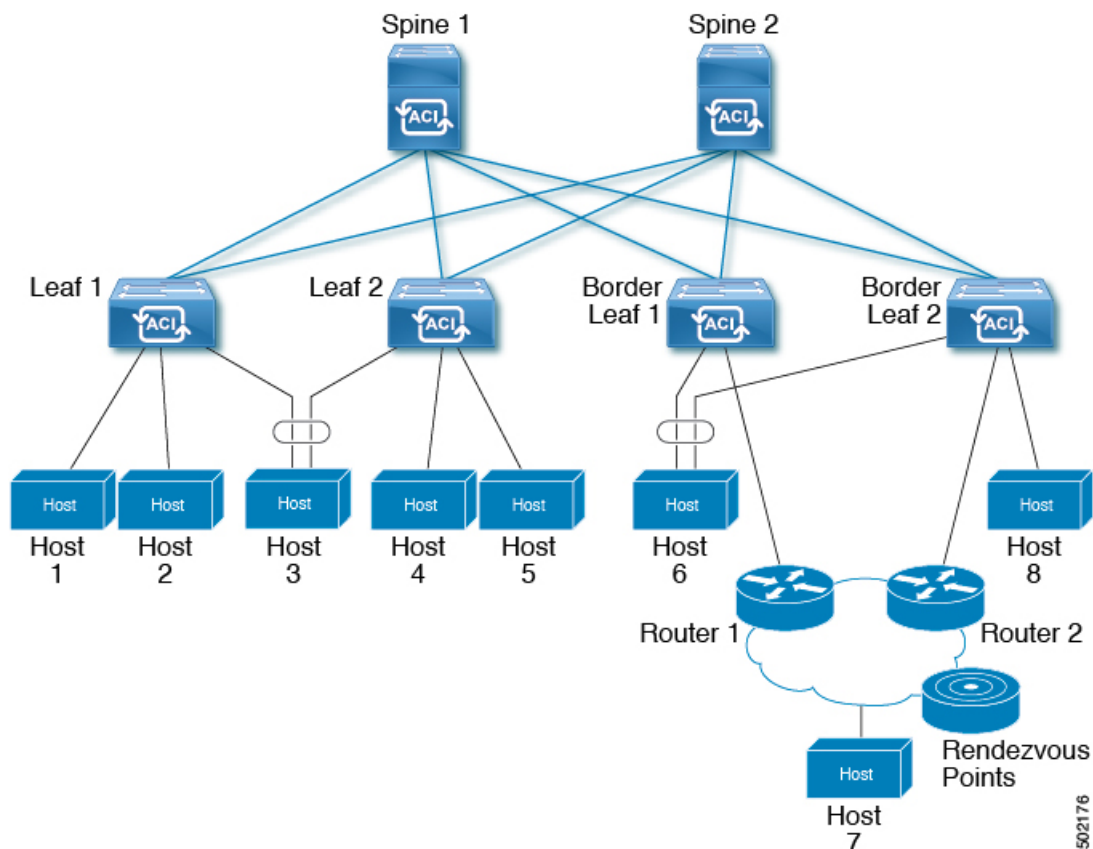
Layer 3 Multicast

In the ACI fabric, most unicast and multicast routing operate together on the same border leaf switches, with the 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 L3 Outs and also with each other.

The following figure shows the border leaf (BL) switches (BL1 and BL2) connecting to routers (R1 and R2) in the multicast cloud. Each virtual routing and forwarding (VRF) in the fabric that requires multicast routing will peer separately with external multicast routers.

Figure 22: 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.
- Layer 3 multicast is not currently supported on VRFs that are configured with a shared L3 Out.
- Any Source Multicast (ASM) and Source-Specific Multicast (SSM) are supported.
- 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.
- 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 preceeding 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.

**Note**

Cisco ACI does not support IP fragmentation. Therefore, when you configure Layer 3 Outside (L3Out) connections to external routers, or multipod connections through an Inter-Pod Network (IPN), it is critical that the MTU is set appropriately on both sides. On some platforms, such as ACI, Cisco NX-OS, and Cisco IOS, the configurable MTU value takes into account the IP headers (resulting in a max packet size to be set as 9216 bytes for ACI and 9000 for NX-OS and IOS). However, other platforms such as IOS-XR configure the MTU value exclusive of packet headers (resulting in a max packet size of 8986 bytes).

For the appropriate MTU values for each platform, see the relevant configuration guides.

Cisco highly recommends 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`.

Configuration Steps for Layer 3 Multicast

The following sections show the configuration steps for layer 3 Multicast. The steps are as follows:

1. Configure PIM options on the tenant VRF.
2. Configure IGMP options for the VRF.
3. Configure an L3 Out for the tenant, enable PIM, and configure the leaf interface.
4. Enable PIM in the desired bridge domains.

Configuring PIM Options for Layer 3 Multicast

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **vrf context** *vrf-name*
4. **[no] ip pim**
5. (Optional) **[no] ip pim auto-rp** {**forward** [**listen**] | **listen** | **mapping-agent-policy** *mapping-agent-policy-name*}
6. (Optional) **[no] ip pim bsr** {**forward** [**listen**] | **listen** | **bsr-policy** *mapping-agent-policy-name*}
7. (Optional) **[no] ip pim fast-convergence**
8. (Optional) **[no] ip pim mtu** *mtu-size*

9. (Optional) **[no] ip pim register-policy** *register-policy-name*
10. (Optional) **[no] ip pim register-rate-limit** *mtu-size*
11. (Optional) **[no] ip pim register-source** *ip-address*
12. (Optional) **[no] ip pim rp-address** *ip-address* [**route-map** *route-map-name*]
13. (Optional) **[no] ip pim sg-expiry-timer** *ip-address* [**sg-list** *route-map-name*]
14. (Optional) **[no] ip pim ssm route-map** *route-map-name*
15. (Optional) **[no] ip pim state-limit** *max-entries* [**reserved** *route-map-name* [*maximum-reserve-state-entries*]]
16. (Optional) **[no] ip pim use-shared-tree-only group-list** *policy-name*
17. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apic1(config)# tenant exampleCorp</code>	Specifies the tenant to be configured.
Step 3	vrf context <i>vrf-name</i> Example: <code>apic1(config-tenant)# vrf context exampleCorp_vrf1</code>	Associates a VRF with the tenant.
Step 4	[no] ip pim Example: <code>apic1(config-tenant-vrf)# ip pim</code>	Configures Protocol Independent Multicast (PIM).
Step 5	(Optional) [no] ip pim auto-rp {forward [listen] listen mapping-agent-policy <i>mapping-agent-policy-name</i> Example: <code>apic1(config-tenant-vrf)# ip pim auto-rp forward listen</code>	Configures PIM auto-RP (Rendezvous Point) options. Auto-RP automates the distribution of group-to-RP mappings in a PIM network. You can choose to forward auto-RP messages, listen to auto-RP messages, or associate a route-map policy for filtering mapping agent messages.
Step 6	(Optional) [no] ip pim bsr {forward [listen] listen bsr-policy <i>mapping-agent-policy-name</i> Example: <code>apic1(config-tenant-vrf)# ip pim bsr forward listen</code>	Configures PIM bootstrap router (BSR) options. BSR performs similarly to auto-RP in that it uses candidate routers for the RP function and for relaying the RP information for a group. RP information is distributed through BSR messages, which are carried within PIM messages. You can choose to forward Bootstrap/Candidate-RP messages, listen to Bootstrap/Candidate-RP messages, or associate a route-map policy for filtering BSR messages.

	Command or Action	Purpose
Step 7	(Optional) [no] ip pim fast-convergence Example: apic1(config-tenant-vrf)# ip pim fast-convergence	Enables the PIM fast convergence feature, which allows the switch to discover unresponsive neighbors more quickly.
Step 8	(Optional) [no] ip pim mtu mtu-size Example: apic1(config-tenant-vrf)# ip pim mtu 1500	Configures the maximum size of a PIM message. The range is 1500 to 65536 bytes.
Step 9	(Optional) [no] ip pim register-policy register-policy-name Example: apic1(config-tenant-vrf)# ip pim register-policy regPolicy1	Specifies the name of a policy for filtering register messages.
Step 10	(Optional) [no] ip pim register-rate-limit mtu-size Example: apic1(config-tenant-vrf)# ip pim register-rate-limit 1024	Specifies a rate limit for PIM data registers. The range is 0 to 65535 packets per second.
Step 11	(Optional) [no] ip pim register-source ip-address Example: apic1(config-tenant-vrf)# ip pim register-source 192.0.20.123	Configures a source IP address for PIM messages.
Step 12	(Optional) [no] ip pim rp-address ip-address [route-map route-map-name] Example: apic1(config-tenant-vrf)# ip pim rp-address 192.0.20.99	Configures a static route processor (RP) address for a multicast group range.
Step 13	(Optional) [no] ip pim sg-expiry-timer ip-address [sg-list route-map-name] Example: apic1(config-tenant-vrf)# ip pim sg-expiry-timer 4096	Configures the (S, G) expiry timer interval for PIM sparse mode (PIM-SM) (S, G) multicast routes. The range is 180 to 604801 seconds. The optional sg-list parameter specifies S,G values to which the timer applies. The default is 4096.
Step 14	(Optional) [no] ip pim ssm route-map route-map-name Example: apic1(config-tenant-vrf)# ip pim ssm route-map SSMRtMap	Configures Source Specific Multicast (SSM), which is an extension of IP multicast in which datagram traffic is forwarded to receivers from only those multicast sources that the receivers have explicitly joined. The route-map policy lists the group prefixes.
Step 15	(Optional) [no] ip pim state-limit max-entries [reserved route-map-name [maximum-reserve-state-entries]] Example: apic1(config-tenant-vrf)# ip pim state-limit 100000 reserved myReservedPolicy 40000	Configures a maximum number of PIM state entries in the current VRF instance. The range is 0 to 4294967295 maximum state entries. You can optionally specify a number of state entries to be reserved for the routes specified in a policy map and you can specify the

	Command or Action	Purpose
		maximum reserved (*, G) and (S, G) entries allowed in this VRF. This number must be less than or equal to the maximum states allowed. The range is from 1 to 4294967295.
Step 16	(Optional) [no] ip pim use-shared-tree-only group-list <i>policy-name</i> Example: <pre>apic1(config-tenant-vrf)# ip pim use-shared-tree-only group-list myGroup1</pre>	Creates the PIM (*, G) state only (where no source state is created). The policy defines the group prefixes where this feature is applied.
Step 17	exit Example: <pre>apic1(config-tenant-vrf)# exit</pre>	Returns to tenant configuration mode.

What to do next

Configure IGMP options for the VRF.

Configuring IGMP Options on the VRF for Layer 3 Multicast

Before you begin

Configure PIM options on the tenant VRF.

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **vrf context** *vrf-name*
4. **[no] ip igmp**
5. **exit**
6. **interface bridge-domain** *bd-name*
7. **[no] ip multicast**
8. **[no] ip igmp allow-v3-asm**
9. **[no] ip igmp fast-leave**
10. **[no] ip igmp group-timeout** *seconds*
11. **[no] ip igmp inherit interface-policy** *policy-name*
12. **[no] ip igmp join-group route-map** *route-map-name*
13. **[no] ip igmp last-member-query-count** *count*
14. **[no] ip igmp last-member-query-response-time** *seconds*
15. **[no] ip igmp querier-timeout** *seconds*
16. **[no] ip igmp query-interval** *seconds*
17. **[no] ip igmp query-max-response-time** *seconds*
18. **[no] ip igmp report-link-local-groups**

19. [no] ip igmp report-policy *policy-name*
20. [no] ip igmp robustness-variable *value*
21. [no] ip igmp snooping
22. [no] ip igmp snooping fast-leave
23. [no] ip igmp snooping last-member-query-interval
24. [no] ip igmp snooping policy *policy-name*
25. [no] ip igmp snooping querier
26. [no] ip igmp snooping query-interval *seconds*
27. [no] ip igmp snooping query-max-response-time *seconds*
28. [no] ip igmp snooping startup-query-count *count*
29. [no] ip igmp snooping startup-query-interval *seconds*
30. [no] ip igmp startup-query-count *count*
31. [no] ip igmp startup-query-interval *seconds*
32. [no] ip igmp state-limit *max-states* [reserved *route-map-name* [*max-reserved-gsg-entries*]]
33. [no] ip igmp static-oif route-map *route-map-name*
34. [no] ip igmp version {v1 | v2 | v3}
35. exit

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: apic1(config)# tenant exampleCorp	Specifies the tenant to be configured.
Step 3	vrf context <i>vrf-name</i> Example: apic1(config-tenant)# vrf context vrf1	Associates a VRF with the tenant.
Step 4	[no] ip igmp Example: apic1(config-tenant-vrf)# ip igmp	Enables Internet Group Management Protocol (IGMP).
Step 5	exit Example: apic1(config-tenant-vrf)# exit	Returns to tenant configuration mode.
Step 6	interface bridge-domain <i>bd-name</i> Example: apic1(config-tenant)# interface bridge-domain exampleCorp_bd1	Enters tenant interface configuration mode to configure the bridge domain.

	Command or Action	Purpose
Step 7	[no] ip multicast Example: <code>apic1(config-tenant-interface)# ip multicast</code>	Enables IP multicast routing on the interface.
Step 8	[no] ip igmp allow-v3-asm Example: <code>apic1(config-tenant-interface)# ip igmp allow-v3-asm</code>	Allows filtering for source addresses in IGMPv3 reports for Any Source Multicast (ASM) groups.
Step 9	[no] ip igmp fast-leave Example: <code>apic1(config-tenant-interface)# ip igmp fast-leave</code>	Enables IP IGMP snooping fast leave processing. This feature supports IGMPv2 hosts that cannot be explicitly tracked because of the host report suppression mechanism of the IGMPv2 protocol. When you enable fast leave, the IGMP software assumes that no more than one host is present on each port.
Step 10	[no] ip igmp group-timeout <i>seconds</i> Example: <code>apic1(config-tenant-interface)# ip igmp group-timeout 260</code>	Sets the group membership timeout for IGMPv2. The range is 3 to 65535 seconds. The default is 260 seconds.
Step 11	[no] ip igmp inherit interface-policy <i>policy-name</i> Example: <code>apic1(config-tenant-interface)# ip igmp inherit interface-policy MyIfPolicy</code>	Associates a IGMP interface policy to this interface.
Step 12	[no] ip igmp join-group route-map <i>route-map-name</i> Example: <code>apic1(config-tenant-interface)# ip igmp join-group route-map MyGroupsRMap</code>	Statically binds one or more multicast groups to the interface. The route-map policy lists the group prefixes, group ranges, and source prefixes.
Step 13	[no] ip igmp last-member-query-count <i>count</i> Example: <code>apic1(config-tenant-interface)# ip igmp last-member-query-count 2</code>	Sets the number of times that the software sends an IGMP query in response to a host leave message. The range is 1 to 5 queries. The default is 2 queries.
Step 14	[no] ip igmp last-member-query-response-time <i>seconds</i> Example: <code>apic1(config-tenant-interface)# ip igmp last-member-query-response-time 1</code>	Sets the query interval waited after sending membership reports before the software deletes the group state. The range is 1 to 25 seconds. The default is 1 second.
Step 15	[no] ip igmp querier-timeout <i>seconds</i> Example: <code>apic1(config-tenant-interface)# ip igmp querier-timeout 255</code>	Sets the number of seconds that the software waits after the previous querier has stopped querying and before it takes over as the querier. The range is 1 to 65535 seconds. The default is 255 seconds.

	Command or Action	Purpose
Step 16	[no] ip igmp query-interval <i>seconds</i> Example: <pre>apic1(config-tenant-interface) # ip igmp query-interval 125</pre>	Sets the frequency at which the software sends IGMP host query messages. You can tune the number of IGMP messages on the network by setting a larger value so that the software sends IGMP queries less often. The range is 1 to 18000 seconds. The default is 125 seconds.
Step 17	[no] ip igmp query-max-response-time <i>seconds</i> Example: <pre>apic1(config-tenant-interface) # ip igmp query-max-response-time 10</pre>	Sets the response time advertised in IGMP queries. You can tune the burstiness of IGMP messages on the network by setting a larger value so that host responses are spread out over a longer time. This value must be less than the query interval. The range is 1 to 25 seconds. The default is 10 seconds.
Step 18	[no] ip igmp report-link-local-groups Example: <pre>apic1(config-tenant-interface) # ip igmp report-link-local-groups</pre>	Enables sending reports for groups in 224.0.0.0/24. Link local addresses are used only by protocols on the local network. Reports are always sent for nonlink local groups. By default, reports are not sent for link local groups.
Step 19	[no] ip igmp report-policy <i>policy-name</i> Example: <pre>apic1(config-tenant-interface) # ip igmp report-policy MyReportPolicy</pre>	Configures an access policy for IGMP reports that is based on a route-map policy.
Step 20	[no] ip igmp robustness-variable <i>value</i> Example: <pre>apic1(config-tenant-interface) # ip igmp robustness-variable 2</pre>	Sets the robustness variable to compensate for packet loss on a congested network. The robustness value is used by the IGMP software to determine the number of times to send messages. You can use a larger value for a lossy network. The range is 1 to 7. The default is 2.
Step 21	[no] ip igmp snooping Example: <pre>apic1(config-tenant-interface) # ip igmp snooping</pre>	Enables IGMP snooping for the interface.
Step 22	[no] ip igmp snooping fast-leave Example: <pre>apic1(config-tenant-interface) # ip igmp snooping fast-leave</pre>	Enables the software to remove the group state when it receives an IGMP Leave report without sending an IGMP query message. This parameter is used for IGMPv2 hosts when no more than one host is present on each port.
Step 23	[no] ip igmp snooping last-member-query-interval Example: <pre>apic1(config-tenant-interface) # ip igmp snooping last-member-query-interval 5</pre>	Sets a time interval in seconds after which the group is removed from the associated port if no hosts respond to an IGMP query message. The range is 1 to 25 seconds. The default is 5 seconds.
Step 24	[no] ip igmp snooping policy <i>policy-name</i> Example: <pre>apic1(config-tenant-interface) # ip igmp snooping policy MySnoopingPolicy</pre>	Associates the bridge domain with an IGMP snooping policy.

	Command or Action	Purpose
Step 25	[no] ip igmp snooping querier Example: <pre>apic1(config-tenant-interface)# ip igmp snooping querier</pre>	Enables an IP IGMP snooping querier, which sends out periodic IGMP queries that trigger IGMP report messages from hosts who want to receive IP multicast traffic. IGMP snooping listens to these IGMP reports to establish appropriate forwarding.
Step 26	[no] ip igmp snooping query-interval <i>seconds</i> Example: <pre>apic1(config-tenant-interface)# ip igmp snooping query-interval 125</pre>	Configures a snooping query interval when you do not enable PIM because multicast traffic does not need to be routed. The range is 1 to 18000 seconds. The default is 125 seconds.
Step 27	[no] ip igmp snooping query-max-response-time <i>seconds</i> Example: <pre>apic1(config-tenant-interface)# ip igmp snooping query-max-response-time 10</pre>	Configures a snooping maximum response time for query messages when you do not enable PIM because multicast traffic does not need to be routed. The range is 1 to 25 seconds. The default is 10 seconds.
Step 28	[no] ip igmp snooping startup-query-count <i>count</i> Example: <pre>apic1(config-tenant-interface)# ip igmp snooping startup-query-count 5</pre>	Configures snooping for a number of queries sent at startup when you do not enable PIM because multicast traffic does not need to be routed. The range is 1 to 10 queries. The default is 5 queries.
Step 29	[no] ip igmp snooping startup-query-interval <i>seconds</i> Example: <pre>apic1(config-tenant-interface)# ip igmp snooping startup-query-interval 15000</pre>	Configures a snooping query interval at startup when you do not enable PIM because multicast traffic does not need to be routed. The range is 1 to 18000 seconds. The default is 15000 seconds.
Step 30	[no] ip igmp startup-query-count <i>count</i> Example: <pre>apic1(config-tenant-interface)# ip igmp startup-query-count 2</pre>	Sets the number of queries sent at startup that are separated by the startup query interval. The range is 1 to 10 queries. The default is 2 queries.
Step 31	[no] ip igmp startup-query-interval <i>seconds</i> Example: <pre>apic1(config-tenant-interface)# ip igmp startup-query-interval 31</pre>	Sets the query interval used when the software starts up. By default, this interval is shorter than the query interval so that the software can establish the group state as quickly as possible. The range is 1 to 18000 seconds. The default is 260 seconds. The default is 31 seconds.
Step 32	[no] ip igmp state-limit <i>max-states</i> [<i>reserved route-map-name</i> [<i>max-reserved-gsg-entries</i>]] Example: <pre>apic1(config-tenant-interface)# ip igmp state-limit 100000 reserved myReservedPolicy 40000</pre>	Configures a per interface limit on the number of mroutes states created as a result of IGMP membership reports (IGMP joins). The range of states allowed is 1 to 4294967295 states. You can optionally specify a number of state entries to be reserved for the routes specified in a policy map and you can specify the maximum reserved (*, G) and (S, G) entries allowed on the interface. The number of reserved states must be less than or equal to the maximum states allowed. The range is from 1 to 4294967295.

	Command or Action	Purpose
Step 33	[no] ip igmp static-oif route-map <i>route-map-name</i> Example: <pre>apic1(config-tenant-interface)# ip igmp static-oif route-map MyOifMap</pre>	Statically binds a multicast group to the outgoing interface (OIF), which is handled by the device hardware. The route map defines the group prefixes where this feature is applied.
Step 34	[no] ip igmp version {v1 v2 v3} Example: <pre>apic1(config-tenant-interface)# ip igmp version v3</pre>	Configures the IGMP version number for the interface. The default version is v2.
Step 35	exit Example: <pre>apic1(config-tenant-interface)# exit</pre>	Returns to tenant configuration mode.

What to do next

Configure an L3 Out for the tenant, enable PIM, and configure the leaf interface.

Configuring an L3 Out for Layer 3 Multicast

Before you begin

- Configure PIM options on the tenant VRF.
- Configure IGMP on the tenant VRF.

SUMMARY STEPS

1. **configure**
2. **tenant *tenant-name***
3. **l3out *l3out-name***
4. **ip pim**
5. **exit**
6. **exit**
7. **leaf *node-id***
8. **interface ethernet *slot/port***
9. **[no] ip igmp allow-v3-asm**
10. **[no] ip igmp fast-leave**
11. **[no] ip igmp group-timeout *seconds***
12. **[no] ip igmp inherit interface-policy *policy-name***
13. **[no] ip igmp join-group route-map *route-map-name***
14. **[no] ip igmp last-member-query-count *count***
15. **[no] ip igmp last-member-query-response-time *seconds***
16. **[no] ip igmp querier-timeout *seconds***
17. **[no] ip igmp query-interval *seconds***

18. **[no] ip igmp query-max-response-time** *seconds*
19. **[no] ip igmp report-link-local-groups**
20. **[no] ip igmp report-policy** *policy-name*
21. **[no] ip igmp robustness-variable** *value*
22. **[no] ip igmp startup-query-count** *count*
23. **[no] ip igmp startup-query-interval** *seconds*
24. **[no] ip igmp state-limit** *max-states* [**reserved** *route-map-name* [*max-reserved-gsg-entries*]]
25. **[no] ip igmp static-oif route-map** *route-map-name*
26. **[no] ip igmp version** {*v1* | *v2* | *v3*}
27. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apic1(config)# tenant exampleCorp</code>	Specifies the tenant to be configured.
Step 3	l3out <i>l3out-name</i> Example: <code>apic1(config-tenant)# l3out exampleCorp_l3out</code>	Configures an L3 Out interface on the tenant.
Step 4	ip pim Example: <code>apic1(config-tenant-l3out)# ip pim</code>	Enables PIM on the interface.
Step 5	exit Example: <code>apic1(config-tenant-l3out)#</code>	Returns to tenant configuration mode.
Step 6	exit Example: <code>apic1(config-tenant)# exit</code>	Returns to global configuration mode.
Step 7	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Enters leaf configuration mode.
Step 8	interface ethernet <i>slot/port</i> Example: <code>apic1(config-leaf)# interface ethernet 1/3</code>	Specifies the interface to be configured.

	Command or Action	Purpose
Step 9	[no] ip igmp allow-v3-asm Example: <code>apic1(config-leaf-if)# ip igmp allow-v3-asm</code>	Allows filtering for source addresses in IGMPv3 reports for Any Source Multicast (ASM) groups.
Step 10	[no] ip igmp fast-leave Example: <code>apic1(config-leaf-if)# ip igmp fast-leave</code>	Enables IP IGMP snooping fast leave processing. This feature supports IGMPv2 hosts that cannot be explicitly tracked because of the host report suppression mechanism of the IGMPv2 protocol. When you enable fast leave, the IGMP software assumes that no more than one host is present on each port.
Step 11	[no] ip igmp group-timeout <i>seconds</i> Example: <code>apic1(config-leaf-if)# ip igmp group-timeout 260</code>	Sets the group membership timeout for IGMPv2. The range is 3 to 65535 seconds. The default is 260 seconds.
Step 12	[no] ip igmp inherit interface-policy <i>policy-name</i> Example: <code>apic1(config-leaf-if)# ip igmp inherit interface-policy MyIfPolicy</code>	Associates a IGMP interface policy to this interface.
Step 13	[no] ip igmp join-group route-map <i>route-map-name</i> Example: <code>apic1(config-leaf-if)# ip igmp join-group route-map MyGroupsRMap</code>	Statically binds one or more multicast groups to the interface. The route-map policy lists the group prefixes, group ranges, and source prefixes.
Step 14	[no] ip igmp last-member-query-count <i>count</i> Example: <code>apic1(config-leaf-if)# ip igmp last-member-query-count 2</code>	Sets the number of times that the software sends an IGMP query in response to a host leave message. The range is 1 to 5 queries. The default is 2 queries.
Step 15	[no] ip igmp last-member-query-response-time <i>seconds</i> Example: <code>apic1(config-leaf-if)# ip igmp last-member-query-response-time 1</code>	Sets the query interval waited after sending membership reports before the software deletes the group state. The range is 1 to 25 seconds. The default is 1 second.
Step 16	[no] ip igmp querier-timeout <i>seconds</i> Example: <code>apic1(config-leaf-if)# ip igmp querier-timeout 255</code>	Sets the number of seconds that the software waits after the previous querier has stopped querying and before it takes over as the querier. The range is 1 to 65535 seconds. The default is 255 seconds.
Step 17	[no] ip igmp query-interval <i>seconds</i> Example: <code>apic1(config-leaf-if)# ip igmp query-interval 125</code>	Sets the frequency at which the software sends IGMP host query messages. You can tune the number of IGMP messages on the network by setting a larger value so that the software sends IGMP queries less often. The range is 1 to 18000 seconds. The default is 125 seconds.

	Command or Action	Purpose
Step 18	[no] ip igmp query-max-response-time <i>seconds</i> Example: <pre>apic1(config-leaf-if)# ip igmp query-max-response-time 10</pre>	Sets the response time advertised in IGMP queries. You can tune the burstiness of IGMP messages on the network by setting a larger value so that host responses are spread out over a longer time. This value must be less than the query interval. The range is 1 to 25 seconds. The default is 10 seconds.
Step 19	[no] ip igmp report-link-local-groups Example: <pre>apic1(config-leaf-if)# ip igmp report-link-local-groups</pre>	Enables sending reports for groups in 224.0.0.0/24. Link local addresses are used only by protocols on the local network. Reports are always sent for nonlink local groups. By default, reports are not sent for link local groups.
Step 20	[no] ip igmp report-policy <i>policy-name</i> Example: <pre>apic1(config-leaf-if)# ip igmp report-policy MyReportPolicy</pre>	Configures an access policy for IGMP reports that is based on a route-map policy.
Step 21	[no] ip igmp robustness-variable <i>value</i> Example: <pre>apic1(config-leaf-if)# ip igmp robustness-variable 2</pre>	Sets the robustness variable to compensate for packet loss on a congested network. The robustness value is used by the IGMP software to determine the number of times to send messages. You can use a larger value for a lossy network. The range is 1 to 7. The default is 2.
Step 22	[no] ip igmp startup-query-count <i>count</i> Example: <pre>apic1(config-leaf-if)# ip igmp startup-query-count 2</pre>	Sets the number of queries sent at startup that are separated by the startup query interval. The range is 1 to 10 queries. The default is 2 queries.
Step 23	[no] ip igmp startup-query-interval <i>seconds</i> Example: <pre>apic1(config-leaf-if)# ip igmp startup-query-interval 31</pre>	Sets the query interval used when the software starts up. By default, this interval is shorter than the query interval so that the software can establish the group state as quickly as possible. The range is 1 to 18000 seconds. The default is 260 seconds. The default is 31 seconds.
Step 24	[no] ip igmp state-limit <i>max-states</i> [reserved <i>route-map-name</i> [<i>max-reserved-gsg-entries</i>]] Example: <pre>apic1(config-leaf-if)# ip igmp state-limit 100000 reserved myReservedPolicy 40000</pre>	Configures a per interface limit on the number of mroutes states created as a result of IGMP membership reports (IGMP joins). The range of states allowed is 1 to 4294967295 states. You can optionally specify a number of state entries to be reserved for the routes specified in a policy map and you can specify the maximum reserved (*, G) and (S, G) entries allowed on the interface. The number of reserved states must be less than or equal to the maximum states allowed. The range is from 1 to 4294967295.
Step 25	[no] ip igmp static-oif route-map <i>route-map-name</i> Example: <pre>apic1(config-leaf-if)# ip igmp static-oif route-map MyOifMap</pre>	Statically binds a multicast group to the outgoing interface (OIF), which is handled by the device hardware. The route map defines the group prefixes where this feature is applied.

Example: Configuring Layer 3 Multicast

	Command or Action	Purpose
Step 26	[no] ip igmp version {v1 v2 v3} Example: apic1(config-leaf-if)# ip igmp version v3	Configures the IGMP version number for the interface. The default version is v2.
Step 27	exit Example: apic1(config-leaf-if)# exit	Returns to tenant configuration mode.

Example: Configuring Layer 3 Multicast

```
# CONFIGURE PIM OPTIONS ON A TENANT VRF

apic1# configure
apic1(config)# tenant exampleCorp
apic1(config-tenant)# vrf context exampleCorp_vrf1
apic1(config-tenant-vrf)# ip pim
apic1(config-tenant-vrf)# ip pim fast-convergence
apic1(config-tenant-vrf)# ip pim bsr forward

# ENABLE AND CONFIGURE IGMP ON THE TENANT VRF AND BRIDGE DOMAIN

apic1(config-tenant-vrf)# ip igmp
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# interface bridge-domain exampleCorp_bd
apic1(config-tenant-interface)# ip multicast
apic1(config-tenant-interface)# ip igmp allow-v3-asm
apic1(config-tenant-interface)# ip igmp fast-leave
apic1(config-tenant-interface)# exit

# CREATE AN L3OUT AND CONFIGURE PIM

apic1(config-tenant)# l3out exampleCorp_l3out
apic1(config-tenant-l3out)# ip pim
apic1(config-tenant-l3out)# exit
apic1(config-tenant)# exit

# CONFIGURE AN EXTERNAL INTERFACE AND CONFIGURE IGMP ON THE INTERFACE

apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/125
apic1(config-leaf-if)# ip igmp fast-leave
apic1(config-leaf-if)# ip-igmp join-group
```

Configuring External-L3 EPGs

External-L3 EPGs are classified under a tenant VRF. In the CLI, an external-l3 EPG is defined in the tenant mode and is deployed to individual nodes. You have the flexibility to place external-l3 EPGs in a select set of nodes instead of all nodes in a VRF.

Each external-L3 EPG can be a producer/consumer of multiple contracts, and each external-L3 EPG has its own QoS policy for DSCP marking and queuing priority within the fabric.

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **external-l3 epg** *epg-name*
4. **vrf member** *vrf-name*
5. **match {ip | ipv6}** *ip-address/masklength*
6. **set qos-class** *class*
7. **set dscp** *dscp-value*
8. **contract consumer** *contract-name*
9. **contract provider** *contract-name*
10. **contract deny** *contract-name*
11. **exit**
12. **exit**
13. **leaf** *node-id*
14. **vrf context tenant** *tenant-name* **vrf** *vrf-name*
15. **external-l3 epg** *epg-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apic1(config)# tenant exampleCorp</code>	Enters the tenant configuration mode.
Step 3	external-l3 epg <i>epg-name</i> Example: <code>apic1(config-tenant)# external-l3 epg epgExtern1</code>	Enters the external-L3 EPG configuration mode.
Step 4	vrf member <i>vrf-name</i> Example: <code>apic1(config-tenant-l3ext-epg)# vrf member v1</code>	Associates the EPG with a VRF.
Step 5	match {ip ipv6} <i>ip-address/masklength</i> Example: <code>apic1(config-tenant-l3ext-epg)# match ip 192.0.20.0/24 apic1(config-tenant-l3ext-epg)# match ipv6 2001::1/64</code>	Creates a rule to match a subnet.

	Command or Action	Purpose
Step 6	set qos-class <i>class</i> Example: <code>apic1(config-tenant-l3ext-epg) # set qos-class level1</code>	Specifies the QOS level for the EPG.
Step 7	set dscp <i>dscp-value</i> Example: <code>apic1(config-tenant-l3ext-epg) # set dscp af31</code>	Specifies the DSCP value for the EPG.
Step 8	contract consumer <i>contract-name</i> Example: <code>apic1(config-tenant-l3ext-epg) # contract consumer cConsumer1</code>	Specifies the consumer contract for the EPG.
Step 9	contract provider <i>contract-name</i> Example: <code>apic1(config-tenant-l3ext-epg) # contract provider cProvider1</code>	Specifies the provider contract for the EPG.
Step 10	contract deny <i>contract-name</i> Example: <code>apic1(config-tenant-l3ext-epg) # contract deny cDeny1</code>	Specifies a deny contract for the EPG.
Step 11	exit Example: <code>apic1(config-tenant-l3ext-epg) # exit</code>	
Step 12	exit Example: <code>apic1(config-tenant) # exit</code>	
Step 13	leaf <i>node-id</i> Example: <code>apic1(config) # leaf 101</code>	Specifies the leaf to be configured.
Step 14	vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-leaf) # vrf context tenant exampleCorp vrf v1</code>	Configures a tenant VRF on the node.
Step 15	external-l3 <i>epg epg-name</i> Example: <code>apic1(config-leaf-vrf) # external-l3 epg epgExtern1</code>	Associates the external layer 3 EPG on the VRF.

Examples

This example shows how to configure an external layer 3 EPG and to deploy the EPG on a leaf.

```
apic1# configure
apic1(config)# tenant exampleCorp

    # CONFIGURE EXTERNAL L3 EPG
apic1(config-tenant)# external-l3 epg epgExtern1
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# match ip 192.0.20.0/24
apic1(config-tenant-l3ext-epg)# match ipv6 2001::1/64
apic1(config-tenant-l3ext-epg)# set qos-class level1
apic1(config-tenant-l3ext-epg)# set dscp af31
apic1(config-tenant-l3ext-epg)# contract consumer cConsumer1
apic1(config-tenant-l3ext-epg)# contract provider cProvider1
apic1(config-tenant-l3ext-epg)# contract deny cDeny1
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# exit

    # DEPLOY EXTERNAL L3 EPG ON A LEAF
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant exampleCorp vrf v1
apic1(config-leaf-vrf)# external-l3 epg epgExtern1
```

Configuring Layer 3 External Connectivity Using the Named Mode

Creating a Named L3Out

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **vrf context** *vrf-name*
4. **l3out** *l3out-name*
5. **vrf member** *vrf-name*
6. **exit**
7. **exit**
8. **leaf** *node-id*
9. **vrf context tenant** *tenant-name* **vrf** *vrf-name* **l3out** *l3out-name*
10. **[no] router-id** *ipv4-address*
11. **[no] {ip | ipv6} route** *ip-prefix/masklen* *next-hop-address* **[preferred]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apic1(config)# tenant exampleCorp</code>	Enters the tenant configuration mode.
Step 3	vrf context <i>vrf-name</i> Example: <code>apic1(config-tenant)# vrf context v1</code>	Associates the tenant with a VRF.
Step 4	l3out <i>l3out-name</i> Example: <code>apic1(config-tenant)# l3out out1</code>	Creates a named L3Out.
Step 5	vrf member <i>vrf-name</i> Example: <code>apic1(config-tenant-l3out)# vrf member v1</code>	Associates the L3Out with the tenant VRF.
Step 6	exit Example: <code>apic1(config-tenant-l3out)# exit</code>	Returns to tenant configuration mode.
Step 7	exit Example: <code>apic1(config-tenant)# exit</code>	Returns to global configuration mode.
Step 8	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	node
Step 9	vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i> l3out <i>l3out-name</i> Example: <code>apic1(config-leaf)# vrf context tenant exampleCorp vrf v1 l3out out1</code>	Configures a tenant VRF on the node.
Step 10	Required: [no] router-id <i>ipv4-address</i> Example: <code>apic1(config-leaf-vrf)# router-id 1.2.3.4</code>	Assigns a router ID for routing protocols running on the VRF.
Step 11	[no] {ip ipv6} route <i>ip-prefix/masklen</i> <i>next-hop-address</i> [preferred]	Configures static route information for the VRF.

	Command or Action	Purpose
	Example: <pre>apic1(config-leaf-vrf) # ip route 21.1.1.1/32 32.1.1.1 apic1(config-leaf-vrf) # ipv6 route 5001::1/128 6002::1</pre>	

Examples

This example shows how to create a named L3Out under the tenant, assign it to the tenant VRF, and deploy it on the border leaf switch.

```
apic1# configure
apic1(config)# tenant exampleCorp
apic1(config-tenant)# vrf context v1
apic1(config-tenant)# l3out out1
apic1(config-tenant-l3out)# vrf member v1
apic1(config-tenant-l3out)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant exampleCorp vrf v1 l3out out1
apic1(config-leaf-vrf)# router-id 1.2.3.4
apic1(config-leaf-vrf)# ip route 21.1.1.1/32 32.1.1.1
```

What to do next

Configure layer 3 interfaces for the named L3Out.

Configuring Layer 3 Interfaces for a Named L3Out

This procedure shows how to configure a layer 3 port interface to a named L3Out. The examples show how to configure a subinterface or SVI to a named L3Out.

- A given interface can be added to multiple L3Outs by providing multiple L3Out names after the **l3out** keyword.
- An SVI can be configured using the **switchport trunk allowed vlan** command under any of the following interface types:
 - **interface Ethernet**
 - **interface port-channel**
 - **interface vpc**

Before you begin

Create a named L3Out.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **interface** *type*
4. **no** **switchport**
5. **vrf member tenant** *tenant-name* **vrf** *vrf-name* **l3out** *l3out-name*
6. **[no] {ip | ipv6} address** *ip-prefix/masklen* **[eui64]** **[secondary]** **[preferred]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: apicl(config)# leaf 101	Specifies the leaf to be configured.
Step 3	interface <i>type</i> Example: apicl(config-leaf)# interface eth 1/20	Specifies a port for the external interface.
Step 4	no switchport Example: apicl(config-leaf-if)# no switchport	Configures the interface as a layer 3 interface, exposing the layer 3 commands in the configuration options.
Step 5	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> l3out <i>l3out-name</i> Example: apicl(config-leaf-if)# vrf member tenant exampleCorp vrf v1 l3out out1	Attaches the interface to the tenant VRF.
Step 6	[no] {ip ipv6} address <i>ip-prefix/masklen</i> [eui64] [secondary] [preferred] Example: apicl(config-leaf-if)# ip address 10.1.1.1/24 apicl(config-leaf-if)# ipv6 address 2001::1/64 preferred	Configures IP addresses on the interface. The specified address can be declared as either: • preferred—The default source address for traffic from the interface. • secondary—The secondary address of the interface. With the optional eui64 keyword, the host can assign itself a 64-bit Extended Unique Identifier (EUI). In this mode, you can also configure ipv6 link-local, mac address, mtu, and other layer 3 properties on the interface.

Examples

This example shows how to assign a layer 3 port to a named L3Out.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface eth 1/20
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member tenant exampleCorp vrf v1 l3out out1
apic1(config-leaf-if)# ip address 10.1.1.1/24
apic1(config-leaf-if)# ipv6 address 2001::1/64 preferred
```

This example shows how to assign a layer 3 subinterface to a named L3Out.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface eth 1/5
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vlan-domain member d1
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 1/5.1000
apic1(config-leaf-if)# vrf member tenant exampleCorp vrf v1 l3out out1
apic1(config-leaf-if)# ip address 10.1.1.1/24
apic1(config-leaf-if)# ipv6 address 2001::1/64 preferred
```

This example shows how to assign a layer 3 SVI to a named L3Out.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface vlan 200
apic1(config-leaf-if)# vrf member tenant exampleCorp vrf v1
apic1(config-leaf-if)# ip address 10.1.1.1/24
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 1/4
apic1(config-leaf-if)# vlan-domain member d1
apic1(config-leaf-if)# switchport trunk allowed vlan 200 tenant t1 external-svi l3out out1
```

Configuring Route Maps for a Named L3Out

- Route-maps are configured under the leaf, VRF mode.
- The following route-maps are created for every named L3Out :
 - Export—Route-map for routes advertised out of a routing protocol enabled on the L3Out. By default, no routes are exported out until you explicitly enable them in the route-map through one or more of **match bridge-domain**, **match prefix-list** and **match community-list** statements.
 - Import—Route-map for routes imported into the routing protocol on the L3Out. By default, all routes are imported. You can control specific routes to be imported by using one or more **match prefix-list** or **match community-list** statements.
 - Shared—Route-map that contains the routes and the contract provider/consumer policy that will be used for leaking the routes from this VRF to any other VRF that has the contract association.

These route-maps are created when you associate a leaf to the L3Out through the **vrf context tenant tenant-name vrf vrf-name l3out l3out-name** command.

- The scope of the route-maps under the named L3Out is always global and is applicable on all nodes where the Named L3Out is deployed.
- All commands under the route-map (such as **match prefix-list**, **match community-list**, **match bridge-domain**) are the same as the route-map configuration for the Basic Mode discussed in the previous sections.

Before you begin

Create a named L3Out.

SUMMARY STEPS

1. **configure**
2. **leaf node-id**
3. **[no] vrf context tenant tenant-name vrf vrf-name l3out l3out-name**
4. **[no] route-map name**
5. **[no] ip prefix-list list-name permit prefix/masklen [le {32 | 128}]**
6. **[no] match prefix-list list-name**
7. **exit**
8. **exit**
9. **[no] route-map name**
10. **[no] ip prefix-list list-name permit prefix/masklen [le {32 | 128}]**
11. **[no] match prefix-list list-name**
12. **set tag name**
13. **exit**
14. **[no] match bridge-domain list-name**
15. **exit**
16. **[no] route-map name**
17. **[no] ip prefix-list list-name permit prefix/masklen [le {32 | 128}]**
18. **[no] match prefix-list list-name**
19. **contract provider name**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	leaf node-id Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.

	Command or Action	Purpose
Step 3	[no] vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i> l3out <i>l3out-name</i> Example: <pre>apic1(config-leaf)# vrf context tenant exampleCorp vrf v1 l3out out1</pre>	Configures a tenant VRF on the node.
Step 4	Required: [no] route-map <i>name</i> Example: <pre>apic1(config-leaf-vrf)# route-map out1_in</pre>	Creates a route-map and enters route-map configuration. This will be the import route-map.
Step 5	Required: [no] ip prefix-list <i>list-name</i> permit <i>prefix/masklen</i> [le {32 128}] Example: <pre>apic1(config-leaf-vrf-route-map)# ip prefix-list p1 permit 15.1.1.0/24</pre>	Creates a prefix-list under the route-map.
Step 6	Required: [no] match prefix-list <i>list-name</i> Example: <pre>apic1(config-leaf-vrf-route-map)# match prefix-list p1</pre>	Matches a prefix-list that has already been created and enters the match mode to configure the route-control profile for the prefix-list.
Step 7	Required: exit Example: <pre>apic1(config-leaf-vrf-route-map-match)# exit</pre>	Returns to route-map configuration mode.
Step 8	Required: exit Example: <pre>apic1(config-leaf-vrf-route-map)# exit</pre>	Returns to leaf VRF configuration mode.
Step 9	Required: [no] route-map <i>name</i> Example: <pre>apic1(config-leaf-vrf)# route-map out1_out</pre>	Creates a route-map and enters route-map configuration. This will be the export route-map.
Step 10	Required: [no] ip prefix-list <i>list-name</i> permit <i>prefix/masklen</i> [le {32 128}] Example: <pre>apic1(config-leaf-vrf-route-map)# ip prefix-list p2 permit 16.1.1.0/24</pre>	Creates a prefix-list under the route-map.
Step 11	Required: [no] match prefix-list <i>list-name</i> Example: <pre>apic1(config-leaf-vrf-route-map)# match prefix-list p2</pre>	Matches a prefix-list that has already been created and enters the match mode to configure the route-control profile for the prefix-list.
Step 12	Required: set tag <i>name</i> Example:	Sets the tag value. The name parameter is an unsigned integer.

	Command or Action	Purpose
	<code>apic1(config-leaf-vrf-route-map-match)# set tag 100</code>	
Step 13	Required: exit Example: <code>apic1(config-leaf-vrf-route-map-match)# exit</code>	Returns to route-map configuration mode.
Step 14	Required: [no] match bridge-domain list-name Example: <code>apic1(config-leaf-vrf-route-map)# match bridge-domain bd1</code>	Matches a bridge domain in order to export its public subnets through the protocol.
Step 15	Required: exit Example: <code>apic1(config-leaf-vrf-route-map-match)# exit</code>	Returns to route-map configuration mode.
Step 16	Required: [no] route-map name Example: <code>apic1(config-leaf-vrf)# route-map out1_shared</code>	Creates a route-map and enters route-map configuration. This will be the shared route-map.
Step 17	Required: [no] ip prefix-list list-name permit prefix/masklen [le {32 128}] Example: <code>apic1(config-leaf-vrf-route-map)# ip prefix-list p3 permit 16.10.1.0/24</code>	Creates a prefix-list under the route-map.
Step 18	Required: [no] match prefix-list list-name Example: <code>apic1(config-leaf-vrf-route-map)# match prefix-list p3</code>	Matches a prefix-list that has already been created and enters the match mode to configure the route-control profile for the prefix-list.
Step 19	Required: contract provider name Example: <code>apic1(config-leaf-vrf-route-map-match)# contract provider default</code>	Adds contract, required to leak routes (matching this prefix-list) from this VRF.

Examples

This example shows how to configure route maps for a named L3Out.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant exampleCorp vrf v1 l3out out1

# CREATE IMPORT ROUTE-MAP
apic1(config-leaf-vrf)# route-map out1_in
apic1(config-leaf-vrf-route-map)# ip prefix-list p1 permit 15.1.1.0/24
```

```

apic1(config-leaf-vrf-route-map)# match prefix-list p1
apic1(config-leaf-vrf-route-map-match)# exit
apic1(config-leaf-vrf-route-map)# exit

# CREATE EXPORT ROUTE-MAP
apic1(config-leaf-vrf)# route-map out1_out
apic1(config-leaf-vrf-route-map)# ip prefix-list p2 permit 16.1.1.0/24
apic1(config-leaf-vrf-route-map)# match prefix-list p2
apic1(config-leaf-vrf-route-map-match)# set tag 100
apic1(config-leaf-vrf-route-map-match)# exit
apic1(config-leaf-vrf-route-map)# match bridge-domain bd1
apic1(config-leaf-vrf-route-map-match)# exit

# CREATE SHARED ROUTE-MAP
apic1(config-leaf-vrf)# route-map out1_shared
apic1(config-leaf-vrf-route-map)# ip prefix-list p3 permit 16.10.1.0/24
apic1(config-leaf-vrf-route-map)# match prefix-list p3
apic1(config-leaf-vrf-route-map-match)# contract provider default

```

Configuring Routing Protocols for a Named L3Out

Configuring BGP for a Named L3Out

- All commands under the BGP neighbor with the exception of **route-map** are identical to those in the Basic Mode of L3Out configuration. The BGP template configuration and the inheritance of the template are identical to the Basic Mode.
- In the Named Mode of L3Out configuration, the route-map is applied at the L3Out level. By associating a neighbor with an L3Out, the route-map is automatically applied on the protocols on the L3Out. For this reason, the route-map option is not applicable and is not available under the BGP Neighbor. For the same reason, the route-map option is not available for OSPF Area and the distribute-list EIGRP option is not available under the interface.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **router bgp** *asn-number*
4. **vrf member** *tenant tenant-name vrf vrf-name*
5. **neighbor** *ip-address[/masklength]* **l3out** *l3out-name*
6. **remote-as** *asn*
7. **allow-self-as-count** *count*
8. **update-source** *ethernet interface-range*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.

	Command or Action	Purpose
Step 2	leaf <i>node-id</i> Example: apicl(config)# leaf 101	Specifies the leaf to be configured.
Step 3	router bgp <i>asn-number</i> Example: apicl(config-leaf)# router bgp 100	Enters BGP policy configuration.
Step 4	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: apicl(config-bgp)# vrf member tenant exampleCorp vrf v100	Specifies the VRF instance to associate with subsequent policy configuration mode commands.
Step 5	neighbor <i>ip-address[/masklength]</i> l3out <i>l3out-name</i> Example: apicl(config-leaf-bgp-vrf)# neighbor 192.0.2.229 l3out out1	Specifies the IP address of the neighbor.
Step 6	remote-as <i>asn</i> Example: apicl(config-leaf-bgp-vrf-neighbor)# remote-as 300	Specifies Autonomous System Number of the neighbor.
Step 7	allow-self-as-count <i>count</i> Example: apicl(config-leaf-bgp-vrf-neighbor)# allow-self-as-count 5	The count can be 1 to 10. The default is 3.
Step 8	update-source ethernet <i>interface-range</i> Example: apicl(config-leaf-bgp-vrf-neighbor)# update-source ethernet 1/3	Update the Source IP for BGP Packets to one of loopback, physical, sub-interface or SVI interfaces..

Examples

This example shows how to configure BGP routing protocol for a named L3Out.

```

apicl# configure
apicl(config)# leaf 101
apicl(config-leaf)# router bgp 100
apicl(config-bgp)# vrf member tenant exampleCorp vrf v1
apicl(config-leaf-bgp-vrf)# neighbor 192.0.2.229 l3out out1
apicl(config-leaf-bgp-vrf-neighbor)# remote-as 300
apicl(config-leaf-bgp-vrf-neighbor)# allow-self-as-count 5
apicl(config-leaf-bgp-vrf-neighbor)# update-source ethernet 1/3

```


Configuring OSPF for a Named L3Out

All commands under the **router ospf default** command, with the exception of **area area-id route-map map-name out**, are identical to those in the Basic Mode of L3Out configuration. The OSPF commands under the interface and the OSPF template inherit commands are also identical to the Basic Mode.

SUMMARY STEPS

1. **configure**
2. **leaf** *node-id*
3. **router ospf default**
4. **vrf member tenant** *tenant-name* **vrf** *vrf-name*
5. **area** *area-id* **l3out** *l3out-name*
6. **area** *area-id* **loopback** *loopback-address*
7. **area** *area-id* **nssa** [**no-redistribution**] [**default-information-originate**]
8. **exit**
9. **exit**
10. **interface** *type*
11. **vlan-domain member** *domain-name*
12. **no** **switchport**
13. **vrf member tenant** *tenant-name* **vrf** *vrf-name* **l3out** *l3out-name*
14. [**no**] {**ip** | **ipv6**} **address** *ip-prefix/masklen* [**eui64**] [**secondary**] [**preferred**]
15. {**ip** | **ipv6**} **router ospf default area** *area-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	router ospf default Example: <code>apic1(config-leaf)# router ospf default</code>	Creates an OSPF routing process and enters OSPF policy configuration.
Step 4	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-leaf-ospf)# vrf member tenant exampleCorp vrf v100</code>	Enables a VRF in the OSPF session.
Step 5	area <i>area-id</i> l3out <i>l3out-name</i> Example:	Enables OSPF in the L3Out.

	Command or Action	Purpose
	<code>apic1(config-leaf-ospf-vrf) # area 0.0.0.1 l3out out1</code>	
Step 6	area <i>area-id</i> loopback <i>loopback-address</i> Example: <code>apic1(config-leaf-ospf-vrf) # area 0.0.0.1 loopback 192.0.20.11</code>	When OSPF is used as a connectivity protocol for BGP, OSPF advertises the loopback address which is used as the source of the BGP session. Note that the loopback IP address and not the loopback ID is used. In this case, a BGP session relying on OSPF will use the same loopback IP address in its update-source command.
Step 7	area <i>area-id</i> nssa [no-redistribution] [default-information-originate] Example: <code>apic1(config-leaf-ospf-vrf) # area 0.0.0.1 nssa</code>	Defines a not-so-stubby area (NSSA).
Step 8	exit Example: <code>apic1(config-leaf-ospf-vrf) # exit</code>	Returns to the OSPF configuration mode.
Step 9	exit Example: <code>apic1(config-leaf-ospf) # exit</code>	Returns to leaf configuration mode.
Step 10	interface <i>type</i> Example: <code>apic1(config-leaf) # interface eth 1/20</code>	Specifies a port for the external interface.
Step 11	vlan-domain member <i>domain-name</i> Example: <code>apic1(config-leaf-if) # vlan-domain member dom1</code>	Assign a VLAN domain to the interface. The VLAN domain must have already been created using the vlan-domain command in the global configuration mode.
Step 12	no switchport Example: <code>apic1(config-leaf-if) # no switchport</code>	Configures the interface as a layer 3 interface, exposing the layer 3 commands in the configuration options.
Step 13	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> l3out <i>l3out-name</i> Example: <code>apic1(config-leaf-if) # vrf member tenant exampleCorp vrf vl l3out out1</code>	Attaches the interface to the tenant VRF.
Step 14	[no] {ip ipv6} address <i>ip-prefix/masklen</i> [eui64] [secondary] [preferred] Example: <code>apic1(config-leaf-if) # ip address 10.1.1.1/24</code>	Configures IP addresses on the interface. The specified address can be declared as either: • preferred—The default source address for traffic from the interface. • secondary—The secondary address of the interface.

	Command or Action	Purpose
	<pre>apic1(config-leaf-if)# ipv6 address 2001::1/64 preferred</pre>	With the optional eui64 keyword, the host can assign itself a 64-bit Extended Unique Identifier (EUI). In this mode, you can also configure ipv6 link-local, mac address, mtu, and other layer 3 properties on the interface.
Step 15	{ip ipv6} router ospf default area area-id Example: <pre>apic1(config-leaf-if)# ip router ospf default area 0.0.0.1</pre>	Creates an OSPF routing process and enters OSPF policy configuration.

Examples

This example shows how to configure OSPF routing protocol for a named L3Out.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant exampleCorp vrf v1
apic1(config-leaf-ospf-vrf)# area 0.0.0.1 l3out out1
apic1(config-leaf-ospf-vrf)# area 0.0.0.1 loopback 192.0.20.11
apic1(config-leaf-ospf-vrf)# area 0.0.0.1 nssa
apic1(config-leaf-ospf-vrf)# exit
apic1(config-leaf-ospf)# exit
apic1(config-leaf)# interface eth 1/20
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member tenant exampleCorp vrf v1 l3out out1
apic1(config-leaf-if)# ip address 10.1.1.1/24
apic1(config-leaf-if)# ip router ospf default area 0.0.0.1
```

Configuring EIGRP for a Named L3Out

All EIGRP commands under vrf mode and interface mode, with the exception of **ip distribute-list**, are identical to those in the Basic Mode of L3Out configuration. This includes the EIGRP **template** and **inherit** commands. The **ip distribute-list** commands are not applicable to the Named Mode of L3Out configuration, as the route-maps are defined at the L3Out level and by associating an interface with the L3Out, the route-map distribute-list is automatically associated. For this reason, **ip distribute-list** is not available in the CLI as an option.

SUMMARY STEPS

1. **configure**
2. **leaf node-id**
3. **router eigrp default**
4. **vrf member tenant tenant-name vrf vrf-name**
5. **autonomous-system asn l3out l3out-name**
6. **exit**
7. **exit**

8. **interface** *type*
9. **vlan-domain member** *domain-name*
10. **no** **switchport**
11. **vrf member tenant** *tenant-name* **vrf** *vrf-name* **l3out** *l3out-name*
12. **[no] {ip | ipv6} address** *ip-prefix/masklen* **[eui64] [secondary] [preferred]**
13. **{ip | ipv6} router eigrp default**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 101</code>	Specifies the leaf to be configured.
Step 3	router eigrp default Example: <code>apic1(config-leaf)# router eigrp default</code>	Enters EIGRP policy configuration.
Step 4	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-eigrp)# vrf member tenant exampleCorp vrf v100</code>	Specifies the VRF instance to associate with subsequent configuration mode commands.
Step 5	autonomous-system <i>asn</i> l3out <i>l3out-name</i> Example: <code>apic1(config-eigrp-vrf)# autonomous-system 500 l3out out1</code>	Enters Autonomous System configuration for EIGRP.
Step 6	exit Example: <code>apic1(config-eigrp-vrf)# exit</code>	Returns to the EIGRP configuration mode.
Step 7	exit Example: <code>apic1(config-eigrp)# exit</code>	Returns to leaf configuration mode.
Step 8	interface <i>type</i> Example: <code>apic1(config-leaf)# interface eth 1/5</code>	Specifies a port for the external interface.

	Command or Action	Purpose
Step 9	vlan-domain member <i>domain-name</i> Example: <code>apic1(config-leaf-if)# vlan-domain member dom1</code>	Assign a VLAN domain to the interface. The VLAN domain must have already been created using the vlan-domain command in the global configuration mode.
Step 10	no switchport Example: <code>apic1(config-leaf-if)# no switchport</code>	Configures the interface as a layer 3 interface, exposing the layer 3 commands in the configuration options.
Step 11	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i> <i>l3out-name</i> Example: <code>apic1(config-leaf-if)# vrf member tenant exampleCorp vrf v1 l3out out1</code>	Attaches the interface to the tenant VRF.
Step 12	[no] {ip ipv6} address <i>ip-prefix/masklen</i> [eui64] [secondary] [preferred] Example: <code>apic1(config-leaf-if)# ip address 10.1.1.1/24</code> <code>apic1(config-leaf-if)# ipv6 address 2001::1/64 preferred</code>	Configures IP addresses on the interface. The specified address can be declared as either: • preferred—The default source address for traffic from the interface. • secondary—The secondary address of the interface. With the optional eui64 keyword, the host can assign itself a 64-bit Extended Unique Identifier (EUI). In this mode, you can also configure ipv6 link-local, mac address, mtu, and other layer 3 properties on the interface.
Step 13	{ip ipv6} router eigrp default Example: <code>apic1(config-leaf-if)# ip router eigrp default</code>	Sets EIGRP policies to default.

Examples

This example shows how to configure EIGRP routing protocol for a named L3Out.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# router eigrp default
apic1(config-eigrp)# vrf member tenant exampleCorp vrf v1
apic1(config-eigrp-vrf)# autonomous-system 500 l3out out1
apic1(config-eigrp-vrf)# exit
apic1(config-eigrp)# exit
apic1(config-leaf)# interface eth 1/5
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member tenant exampleCorp vrf v1 l3out out1
apic1(config-leaf-if)# ip address 10.1.1.1/24
apic1(config-leaf-if)# ip router eigrp default
```

Configuring External-L3 EPGs for a Named L3Out

External-L3 EPGs are classified under a tenant VRF.

All commands under the **config-tenant-l3ext-epg** mode are identical to those in the Basic Mode of L3Out configuration with the following differences:

- The VRF is automatically associated with the EPG. The L3Out associates with a VRF and the EPG associates with the L3Out.
- The **external-l3 epg** command is not available under the leaf **vrf context tenant tenant-name vrf vrf-name l3out l3out-name** command, as this configuration is not applicable for Named L3Outs. The **external-l3 epg** is automatically deployed on the leaf, when the **external-l3 epg** is created within a named L3Out and a leaf is associated with the same L3Out through the **vrf context tenant tenant-name vrf vrf-name l3out l3out-name** command.

SUMMARY STEPS

1. **configure**
2. **tenant tenant-name**
3. **external-l3 epg epg-name l3out l3out-name**
4. **match {ip | ipv6} ip-address/masklength**
5. **contract consumer contract-name**
6. **contract provider contract-name**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	tenant tenant-name Example: apic1(config)# tenant exampleCorp	Enters the tenant configuration mode.
Step 3	external-l3 epg epg-name l3out l3out-name Example: apic1(config-tenant)# external-l3 epg epg1 l3out out1	Enters the external-l3 EPG configuration mode.
Step 4	match {ip ipv6} ip-address/masklength Example: apic1(config-tenant-l3ext-epg)# match ip 192.0.20.0/24 apic1(config-tenant-l3ext-epg)# match ipv6 2001::1/64	Creates a rule to match a subnet.

	Command or Action	Purpose
Step 5	contract consumer <i>contract-name</i> Example: <pre>apic1(config-tenant-l3ext-epg) # contract consumer cConsumer1</pre>	Specifies the consumer contract for the EPG.
Step 6	contract provider <i>contract-name</i> Example: <pre>apic1(config-tenant-l3ext-epg) # contract provider cProvider1</pre>	Specifies the provider contract for the EPG.

Examples

This example shows how to configure an external layer 3 EPG for a named L3Out.

```
apic1# configure
apic1(config)# tenant exampleCorp
apic1(config-tenant)# external-l3 epg epg1 l3out out1
apic1(config-tenant-l3ext-epg) # match ip 192.0.20.0/24
apic1(config-tenant-l3ext-epg) # match ipv6 2001::1/64
apic1(config-tenant-l3ext-epg) # contract consumer cConsumer1
apic1(config-tenant-l3ext-epg) # contract provider cProvider1
```

IPv6 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)

Configuring a Tenant, VRF, and Bridge Domain with IPv6 Neighbor Discovery on the Bridge Domain Using the NX-OS Style CLI

Step 1 Configure an IPv6 neighbor discovery interface policy and assign it to a bridge domain:

a) Create an IPv6 neighbor discovery interface policy:

Example:

```
apic1(config)# tenant ExampleCorp
apic1(config-tenant)# template ipv6 nd policy NDPol001
apic1(config-tenant-template-ipv6-nd)# ipv6 nd mtu 1500
```

b) Create a VRF and bridge domain:

Example:

```
apic1(config-tenant)# vrf context pvn1
```



```
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# bridge-domain bd1
apic1(config-tenant-bd)# vrf member pvn1
apic1(config-tenant-bd)# exit
```

- c) Assign an IPv6 neighbor discovery policy to the bridge domain:

Example:

```
apic1(config-tenant)# interface bridge-domain bd1
apic1(config-tenant-interface)# ipv6 nd policy NDPol001
apic1(config-tenant-interface)#exit
```

- Step 2** Configure an IPV6 bridge domain subnet and neighbor discovery prefix policy on the subnet:

Example:

```
apic1(config-tenant)# interface bridge-domain bd1
apic1(config-tenant-interface)# ipv6 address 34::1/64
apic1(config-tenant-interface)# ipv6 address 33::1/64
apic1(config-tenant-interface)# ipv6 nd prefix 34::1/64 1000 1000
apic1(config-tenant-interface)# ipv6 nd prefix 33::1/64 4294967295 4294967295
```

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 NX-OS Style CLI

This example configures an IPv6 neighbor discovery interface policy, and assigns it to a Layer 3 interface. Next, it configures an IPv6 Layer 3 Out interface, neighbor discovery prefix policy, and associates the neighbor discovery policy to the interface.

Procedure

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	tenant <i>tenant_name</i> Example: apic1(config)# tenant ExampleCorp	Creates a tenant and enters the tenant mode.

	Command or Action	Purpose
	<code>apic1(config-tenant) #</code>	
Step 3	template ipv6 nd policy <i>policy_name</i> Example: <code>apic1(config-tenant) # template ipv6 nd policy NDPo1001</code>	Creates an IPv6 ND policy.
Step 4	ipv6 nd mtu <i>mtu value</i> Example: <code>apic1(config-tenant-template-ipv6-nd) # ipv6 nd mtu 1500</code> <code>apic1(config-tenant-template-ipv6) # exit</code> <code>apic1(config-tenant-template) # exit</code> <code>apic1(config-tenant) #</code>	Assigns an MTU value to the IPv6 ND policy.
Step 5	vrf context <i>VRF_name</i> Example: <code>apic1(config-tenant) # vrf context pvn1</code> <code>apic1(config-tenant-vrf) # exit</code>	Creates a VRF.
Step 6	l3out <i>VRF_name</i> Example: <code>apic1(config-tenant) # l3out l3extOut001</code>	Creates a Layer 3 Out.
Step 7	vrf member <i>VRF_name</i> Example: <code>apic1(config-tenant-l3out) # vrf member pvn1</code> <code>apic1(config-tenant-l3out) # exit</code>	Associates the VRF with the Layer 3 Out.
Step 8	external-l3 epg instp <i>l3out l3extOut001</i> Example: <code>apic1(config-tenant) # external-l3 epg instp l3out l3extOut001</code> <code>apic1(config-tenant-l3ext-epg) # vrf member pvn1</code> <code>apic1(config-tenant-l3ext-epg) # exit</code>	Assigns the Layer 3 Out and the VRF to a Layer 3 interface.
Step 9	leaf <i>2011</i> Example:	Enters the leaf switch mode.

	Command or Action	Purpose
	<code>apic1(config)# leaf 2011</code>	
Step 10	vrf context tenant <i>ExampleCorp</i> vrf pvn1 l3out l3extOut001 Example: <code>apic1(config-leaf)# vrf context tenant ExampleCorp vrf pvn1 l3out l3extOut001 apic1(config-leaf-vrf)# exit</code>	Associates the VRF to the leaf switch.
Step 11	int eth 1/1 Example: <code>apic1(config-leaf)# int eth 1/1 apic1(config-leaf-if)#</code>	Enters the interface mode.
Step 12	vrf member tenant <i>ExampleCorp</i> vrf pvn1 l3out l3extOut001 Example: <code>apic1(config-leaf-if)# vrf member tenant ExampleCorp vrf pvn1 l3out l3extOut001</code>	Specifies the associated Tenant, VRF, Layer 3 Out in the interface.
Step 13	ipv6 address 2001:20:21:22::2/64 preferred Example: <code>apic1(config-leaf-if)# ipv6 address 2001:20:21:22::2/64 preferred</code>	Specifies the primary or preferred IPv6 address.
Step 14	ipv6 nd prefix 2001:20:21:22::2/64 1000 1000 Example: <code>apic1(config-leaf-if)# ipv6 nd prefix 2001:20:21:22::2/64 1000 1000</code>	Configures the IPv6 ND prefix policy under the Layer 3 interface.
Step 15	inherit ipv6 nd NDPol001 Example: <code>apic1(config-leaf-if)# inherit ipv6 nd NDPol001 apic1(config-leaf-if)# exit apic1(config-leaf)# exit</code>	Configures the ND policy under the Layer 3 interface.

The configuration is complete.

Microsoft NLB

Configuring Microsoft NLB in Unicast Mode Using the NX-OS Style CLI

This task configures Microsoft NLB to flood all of the ports in the bridge domain.

Before you begin

Have the following information available before proceeding with these procedures:

- Microsoft NLB cluster VIP
- Microsoft NLB cluster MAC address

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **application** *app-profile-name*
4. **epg** *epg-name*
5. **[no] endpoint {ip | ipv6} ip-address eplb mode mode-uc mac mac-address**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: apicl (config)# tenant tenant1	Creates a tenant if it does not exist or enters tenant configuration mode.
Step 3	application <i>app-profile-name</i> Example: apicl (config-tenant)# application app1	Creates an application profile if it doesn't exist or enters application profile configuration mode.
Step 4	epg <i>epg-name</i> Example: apicl (config-tenant-app)# epg epg1	Creates an EPG if it doesn't exist or enters EPG configuration mode.
Step 5	[no] endpoint {ip ipv6} ip-address eplb mode mode-uc mac mac-address Example:	Configures Microsoft NLB in unicast mode, where: • <i>ip-address</i> is the Microsoft NLB cluster VIP.

	Command or Action	Purpose
	<code>apic1 (config-tenant-app-epg) # endpoint ip 192.0.2.2/32 eplb mode mode-uc mac 03:BF:01:02:03:04</code>	• <i>mac-address</i> is the Microsoft NLB cluster MAC address.

Configuring Microsoft NLB in Multicast Mode Using the NX-OS Style CLI

This task configures Microsoft NLB to flood only on certain ports in the bridge domain.

Before you begin

Have the following information available before proceeding with these procedures:

- Microsoft NLB cluster VIP
- Microsoft NLB cluster MAC address

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **application** *app-profile-name*
4. **epg** *epg-name*
5. **[no] endpoint {ip | ipv6} ip-address eplb mode mode-mcast--static mac mac-address**
6. **[no] nld static-group mac-address leaf leaf-num interface {ethernet slot/port | port-channel port-channel-name} vlan portEncapVlan**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <code>apic1 (config) # tenant tenant1</code>	Creates a tenant if it does not exist or enters tenant configuration mode.
Step 3	application <i>app-profile-name</i> Example: <code>apic1 (config-tenant) # application app1</code>	Creates an application profile if it doesn't exist or enters application profile configuration mode.
Step 4	epg <i>epg-name</i> Example: <code>apic1 (config-tenant-app) # epg epg1</code>	Creates an EPG if it doesn't exist or enters EPG configuration mode.
Step 5	[no] endpoint {ip ipv6} ip-address eplb mode mode-mcast--static mac mac-address	Configures Microsoft NLB in static multicast mode, where:

	Command or Action	Purpose
	Example: <pre>apicl (config-tenant-app-epg) # endpoint ip 192.0.2.2/32 epnlb mode mode-mcast--static mac 03:BF:01:02:03:04</pre>	• <i>ip-address</i> is the Microsoft NLB cluster VIP. • <i>mac-address</i> is the Microsoft NLB cluster MAC address.
Step 6	[no] nld static-group mac-address leaf leaf-num interface {ethernet slot/port port-channel port-channel-name} vlan portEncapVlan Example: <pre>apicl (config-tenant-app-epg) # nld static-group 03:BF:01:02:03:04 leaf 102 interface ethernet 1/12 vlan 19</pre>	Adds Microsoft NLB multicast VMAC to the EPG ports where the Microsoft NLB servers are connected, where: • <i>mac-address</i> is the Microsoft NLB cluster MAC address that you entered in Step 5, on page 297. • <i>leaf-num</i> is the leaf switch that contains the interface to be added or removed. • <i>port-channel-name</i> is the name of the port-channel, when the port-channel option is used. • <i>portEncapVlan</i> is the encapsulation VLAN for the static member of the application EPG.

Configuring Microsoft NLB in IGMP Mode Using the NX-OS Style CLI

This task configures Microsoft NLB to flood only on certain ports in the bridge domain.

Before you begin

Have the following information available before proceeding with these procedures:

- Microsoft NLB cluster VIP
- Microsoft NLB cluster MAC address

SUMMARY STEPS

1. **configure**
2. **tenant** *tenant-name*
3. **application** *app-profile-name*
4. **epg** *epg-name*
5. **[no] endpoint {ip | ipv6} ip-address epnlb mode mode-mcast-igmp group multicast-IP-address**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <pre>apicl# configure</pre>	Enters configuration mode.

	Command or Action	Purpose
Step 2	tenant <i>tenant-name</i> Example: <code>apic1 (config)# tenant tenant1</code>	Creates a tenant if it does not exist or enters tenant configuration mode.
Step 3	application <i>app-profile-name</i> Example: <code>apic1 (config-tenant)# application app1</code>	Creates an application profile if it doesn't exist or enters application profile configuration mode.
Step 4	epg <i>epg-name</i> Example: <code>apic1 (config-tenant-app)# epg epg1</code>	Creates an EPG if it doesn't exist or enters EPG configuration mode.
Step 5	[no] endpoint {ip ipv6} ip-address eplb mode mode-mcast-igmp group multicast-IP-address Example: <code>apic1 (config-tenant-app-epg)# endpoint ip 192.0.2.2/32 eplb mode mode-mcast-igmp group 1.3.5.7</code>	Configures Microsoft NLB in IGMP mode, where: • <i>ip-address</i> is the Microsoft NLB cluster VIP. • <i>multicast-IP-address</i> is the multicast IP for the NLB endpoint group.

MLD Snooping

Configuring and Assigning an MLD Snooping Policy to a Bridge Domain using the NX-OS Style CLI

Before you begin

- Create the tenant that will consume the MLD Snooping policy.
- Create the bridge domain for the tenant, where you will attach the MLD Snooping policy.

SUMMARY STEPS

1. **configure terminal**
2. **tenant** *tenant-name*
3. **template ipv6 mld snooping policy** *policy-name*
4. **[no] ipv6 mld snooping**
5. **[no] ipv6 mld snooping fast-leave**
6. **[no] ipv6 mld snooping querier**
7. **ipv6 mld snooping last-member-query-interval** *parameter*
8. **ipv6 mld snooping query-interval** *parameter*
9. **ipv6 mld snooping query-max-response-time** *parameter*
10. **ipv6 mld snooping startup-query-count** *parameter*

11. `ipv6 mld snooping startup-query-interval` *parameter*
12. `exit`
13. `interface bridge-domain` *bridge-domain-name*
14. `ipv6 address` *sub-bits/prefix-length* `snooping-querier`
15. `ipv6 mld snooping policy` *policy-name*
16. `exit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>apic1# configure terminal apic1(config)#</pre>	Enters configuration mode.
Step 2	tenant <i>tenant-name</i> Example: <pre>apic1(config)# tenant tn1 apic1(config-tenant)#</pre>	Creates a tenant or enters tenant configuration mode.
Step 3	template ipv6 mld snooping policy <i>policy-name</i> Example: <pre>apic1(config-tenant)# template ipv6 mld snooping policy mldPolicy1 apic1(config-tenant-template-ip-mld-snooping)#</pre>	Creates an MLD snooping policy. The example NX-OS style CLI sequence creates an MLD snooping policy named mldPolicy1.
Step 4	[no] ipv6 mld snooping Example: <pre>apic1(config-tenant-template-ip-mld-snooping)# ipv6 mld snooping apic1(config-tenant-template-ip-mld-snooping)# no ipv6 mld snooping</pre>	Enables or disables the admin state of the MLD snoop policy. The default state is disabled.
Step 5	[no] ipv6 mld snooping fast-leave Example: <pre>apic1(config-tenant-template-ip-mld-snooping)# ipv6 mld snooping fast-leave apic1(config-tenant-template-ip-mld-snooping)# no ipv6 mld snooping fast-leave</pre>	Enables or disables IPv6 MLD snooping fast-leave processing.
Step 6	[no] ipv6 mld snooping querier Example: <pre>apic1(config-tenant-template-ip-mld-snooping)#</pre>	Enables or disables IPv6 MLD snooping querier processing. For the enabling querier option to be effectively enabled on the assigned policy, you must also enable the querier option in the subnets assigned to the bridge

	Command or Action	Purpose
	<pre> ipv6 mld snooping querier apic1 (config-tenant-template-ip-mld-snooping) # no ipv6 mld snooping querier </pre>	domains to which the policy is applied, as described in Step 14, on page 302 .
Step 7	ipv6 mld snooping last-member-query-interval <i>parameter</i> Example: <pre> apic1 (config-tenant-template-ip-mld-snooping) # ipv6 mld snooping last-member-query-interval 25 </pre>	Changes the IPv6 MLD snooping last member query interval parameter. The example NX-OS style CLI sequence changes the IPv6 MLD snooping last member query interval parameter to 25 seconds. Valid options are 1-25. The default is 1 second.
Step 8	ipv6 mld snooping query-interval <i>parameter</i> Example: <pre> apic1 (config-tenant-template-ip-mld-snooping) # ipv6 mld snooping query-interval 300 </pre>	Changes the IPv6 MLD snooping query interval parameter. The example NX-OS style CLI sequence changes the IPv6 MLD snooping query interval parameter to 300 seconds. Valid options are 1-18000. The default is 125 seconds.
Step 9	ipv6 mld snooping query-max-response-time <i>parameter</i> Example: <pre> apic1 (config-tenant-template-ip-mld-snooping) # ipv6 mld snooping query-max-response-time 25 </pre>	Changes the IPv6 MLD snooping max query response time. The example NX-OS style CLI sequence changes the IPv6 MLD snooping max query response time to 25 seconds. Valid options are 1-25. The default is 10 seconds.
Step 10	ipv6 mld snooping startup-query-count <i>parameter</i> Example: <pre> apic1 (config-tenant-template-ip-mld-snooping) # ipv6 mld snooping startup-query-count 10 </pre>	Changes the IPv6 MLD snooping number of initial queries to send. The example NX-OS style CLI sequence changes the IPv6 MLD snooping number of initial queries to send to 10. Valid options are 1-10. The default is 2.
Step 11	ipv6 mld snooping startup-query-interval <i>parameter</i> Example: <pre> apic1 (config-tenant-template-ip-mld-snooping) # ipv6 mld snooping startup-query-interval 300 </pre>	Changes the IPv6 MLD snooping time for sending initial queries. The example NX-OS style CLI sequence changes the IPv6 MLD snooping time for sending initial queries to 300 seconds. Valid options are 1-18000. The default is 31 seconds.
Step 12	exit Example: <pre> apic1 (config-tenant-template-ip-mld-snooping) # exit apic1 (config-tenant) # </pre>	Returns to configure mode.
Step 13	interface bridge-domain <i>bridge-domain-name</i> Example: <pre> apic1 (config-tenant) # interface bridge-domain bd1 apic1 (config-tenant-interface) # </pre>	Configures the interface bridge-domain. The example NX-OS style CLI sequence configures the interface bridge-domain named bd1.

	Command or Action	Purpose
Step 14	ipv6 address <i>sub-bits/prefix-length</i> snooping-querier Example: <pre>apic1(config-tenant-interface) # ipv6 address 2000::5/64 snooping-querier</pre>	Configures the bridge domain as switch-querier. This will enable the querier option in the subnet assigned to the bridge domain where the policy is applied.
Step 15	ipv6 mld snooping policy <i>policy-name</i> Example: <pre>apic1(config-tenant-interface) # ipv6 mld snooping policy mldPolicy1</pre>	Associates the bridge domain with an MLD snooping policy. The example NX-OS style CLI sequence associates the bridge domain with an MLD snooping policy named mldPolicy1.
Step 16	exit Example: <pre>apic1(config-tenant-interface) # exit apic1(config-tenant) #</pre>	Returns to configure mode.

Configuring HSRP

Configuring HSRP in Cisco APIC Using Inline Parameters in NX-OS Style CLI

HSRP is enabled when the leaf switch is configured.

Before you begin

- The tenant and VRF 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.

SUMMARY STEPS

1. **configure**
2. Configure HSRP by creating inline parameters.

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <pre>apic1# configure</pre>	Enters configuration mode.

	Command or Action	Purpose
Step 2	Configure HSRP by creating inline parameters. Example: <pre> apic1(config)# leaf 101 apic1(config-leaf)# interface ethernet 1/17 apic1(config-leaf-if)# hsrp version 1 apic1(config-leaf-if)# hsrp use-bia apic1(config-leaf-if)# hsrp delay minimum 30 apic1(config-leaf-if)# hsrp delay reload 30 apic1(config-leaf-if)# hsrp 10 ipv4 apic1(config-if-hsrp)# ip 182.16.1.2 apic1(config-if-hsrp)# ip 182.16.1.3 secondary apic1(config-if-hsrp)# ip 182.16.1.4 secondary apic1(config-if-hsrp)# mac-address 5000.1000.1060 apic1(config-if-hsrp)# timers 5 18 apic1(config-if-hsrp)# priority 100 apic1(config-if-hsrp)# preempt apic1(config-if-hsrp)# preempt delay minimum 60 apic1(config-if-hsrp)# preempt delay reload 60 apic1(config-if-hsrp)# preempt delay sync 60 apic1(config-if-hsrp)# authentication none apic1(config-if-hsrp)# authentication simple apic1(config-if-hsrp)# authentication md5 apic1(config-if-hsrp)# authentication-key <mypassword> apic1(config-if-hsrp)# authentication-key-timeout <timeout> </pre>	

Configuring HSRP in Cisco APIC Using Template and Policy in NX-OS Style CLI

HSRP is enabled when the leaf switch is configured.

Before you begin

- The tenant and VRF 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.

SUMMARY STEPS

1. **configure**
2. Configure HSRP policy templates.
3. Use the configured policy templates

DETAILED STEPS

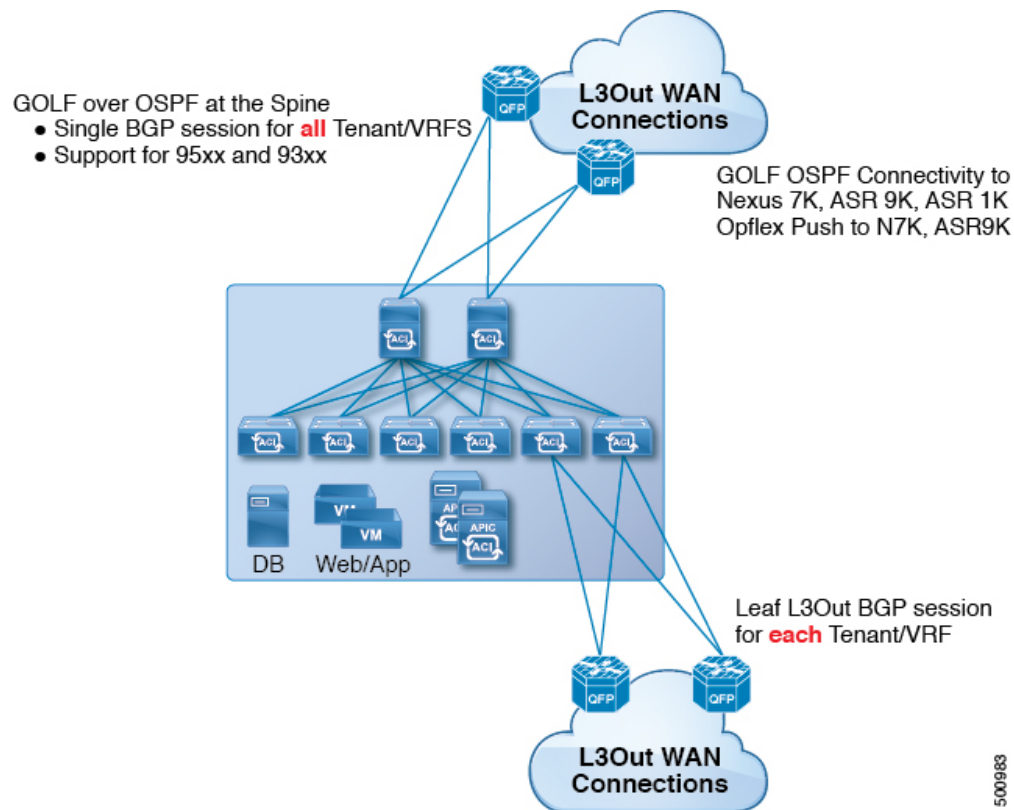
	Command or Action	Purpose
Step 1	configure Example: <code>apicl# configure</code>	Enters configuration mode.
Step 2	Configure HSRP policy templates. Example: <pre> apicl(config)# leaf 101 apicl(config-leaf)# template hsrp interface-policy hsrp-intfPol1 tenant t9 apicl(config-template-hsrp-if-pol)# hsrp use-bia apicl(config-template-hsrp-if-pol)# hsrp delay minimum 30 apicl(config-template-hsrp-if-pol)# hsrp delay reload 30 apicl(config)# leaf 101 apicl(config-leaf)# template hsrp group-policy hsrp-groupPol1 tenant t9 apicl(config-template-hsrp-group-pol)# timers 5 18 apicl(config-template-hsrp-group-pol)# priority 100 apicl(config-template-hsrp-group-pol)# preempt apicl(config-template-hsrp-group-pol)# preempt delay minimum 60 apicl(config-template-hsrp-group-pol)# preempt delay reload 60 apicl(config-template-hsrp-group-pol)# preempt delay sync 60 </pre>	
Step 3	Use the configured policy templates Example: <pre> apicl(config)# leaf 101 apicl(config-leaf)# interface ethernet 1/17 apicl(config-leaf-if)# hsrp version 1 apicl(config-leaf-if)# inherit hsrp interface-policy hsrp-intfPol1 apicl(config-leaf-if)# hsrp 10 ipv4 apicl(config-if-hsrp)# ip 182.16.1.2 apicl(config-if-hsrp)# ip 182.16.1.3 secondary apicl(config-if-hsrp)# ip 182.16.1.4 secondary apicl(config-if-hsrp)# mac-address 5000.1000.1060 apicl(config-if-hsrp)# inherit hsrp group-policy hsrp-groupPol1 </pre>	

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 23: 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`.

Guidelines and Limitations

Observe the following GOLF guidelines and limitations:

- GOLF routers must advertise at least one route to Cisco ACI in order to accept traffic. No tunnel is created between leaf switches and the external routers until Cisco ACI receives a route from the external routers.
- All Cisco Nexus 9000 Series ACI-mode switches and all of the Cisco Nexus 9500 platform ACI-mode switch line cards and fabric modules support GOLF. With Cisco APIC, release 3.1(x) and higher, this includes the N9K-C9364C switch.
- At this time, only a single GOLF provider policy can be deployed on spine switch interfaces for the whole fabric.
- Up to APIC release 2.0(2), GOLF is not supported with multipod. 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.
- When configuring GOLF on a spine switch, wait for the control plane to converge before configuring GOLF on another spine switch.
- A spine switch can be added to multiple provider GOLF outside networks (GOLF L3Outs), but the provider labels have to be different for each GOLF L3Out. Also, in this case, the OSPF Area has to be different on each of the `L3extOuts` and use different loopback addresses.
- The BGP EVPN session in the matching provider `L3Out` in the `infra` tenant advertises the tenant routes defined in this `L3extOut`.
- When deploying three GOLF Outs, if only 1 has a provider/consumer label for GOLF, and 0/0 export aggregation, APIC will export all routes. This is the same as existing `L3extOut` on leaf switches for tenants.
- If there is direct peering between a spine switch and a data center interconnect (DCI) router, the transit routes from leaf switches to the ASR have the next hop as the PTEP of the leaf switch. In this case, define a static route on the ASR for the TEP range of that ACI pod. Also, if the DCI is dual-homed to the same

pod, then the precedence (administrative distance) of the static route should be the same as the route received through the other link.

- The default `bgpPeerPfxPol` policy restricts routes to 20,000. For ACI WAN Interconnect peers, increase this as needed.
- In a deployment scenario where there are two `L3extOutS` on one spine switch, and one of them has the provider label `prov1` and peers with the DCI 1, the second `L3extOut` peers with DCI 2 with provider label `prov2`. If the tenant VRF has a consumer label pointing to any 1 of the provider labels (either `prov1` or `prov2`), the tenant route will be sent out both DCI 1 and DCI 2.
- When aggregating GOLF OpFlex VRFs, the leaking of routes cannot occur in the ACI fabric or on the GOLF device between the GOLF OpFlex VRF and any other VRF in the system. An external device (not the GOLF router) must be used for the VRF leaking.

**Note**

Cisco ACI does not support IP fragmentation. Therefore, when you configure Layer 3 Outside (L3Out) connections to external routers, or multipod connections through an Inter-Pod Network (IPN), it is critical that the MTU is set appropriately on both sides. On some platforms, such as ACI, Cisco NX-OS, and Cisco IOS, the configurable MTU value takes into account the IP headers (resulting in a max packet size to be set as 9216 bytes for ACI and 9000 for NX-OS and IOS). However, other platforms such as IOS-XR configure the MTU value exclusive of packet headers (resulting in a max packet size of 8986 bytes).

For the appropriate MTU values for each platform, see the relevant configuration guides.

Cisco highly recommends 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`.

Configuration Tasks to Configure Cisco ACI GOLF Services Using the NX-OS Style CLI

Perform the following tasks to configure GOLF services (using the BGP EVPN protocol), with the NX-OS style CLI:

- Configure the infra tenant for BGP EVPN, including the VLAN domain, VRF, Interface IP addressing, and OSPF.
- Configure BGP on the spine node to support BGP EVPN.
- Configure a tenant for BGP EVPN.
- Configure the BGP EVPN route target, route map, and prefix-epg for the tenant.
- Configure BGP address-families to enable distributing BGP EVPN type-2 (MAC-IP) host routes to the DCIG, with the **host-rt-enable** command.

Configuring a Spine and the Infra Tenant for BGP EVPN, Using the NX-OS Style CLI

This task describes how to configure the infra tenant for BGP EVPN, including the VLAN domain, VRF, Interface IP addressing, and OSPF in the following steps:

Procedure

	Command or Action	Purpose
Step 1	configure	Enters global configuration mode. <code>apic1# configure</code>
Step 2	vlan-domain <i>vlan-domain-name</i> dynamic	Creates a VLAN domain. <code>apic1(config)# vlan-domain evpn-dom dynamic</code>
Step 3	spine <i>spine-name</i>	Creates the spine or enters spine configuration mode. <code>apic1(config)# spine 111</code>
Step 4	vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i>	Associates the VRF with the tenant. <code>apic1(config-spine)# vrf context tenant infra vrf overlay-1</code>
Step 5	router-id <i>A.B.C.D</i>	Configures the router ID for the VRF. <code>apic1(config-spine-vrf)# router-id 10.10.3.3</code>
Step 6	exit	Returns to spine configuration mode. <code>apic1(config-spine-vrf)# exit</code>
Step 7	interface ethernet <i>slot/port</i>	Configures an interface for a spine node. <code>apic1(config-spine)# interface ethernet 1/33</code>
Step 8	vlan-domain member <i>vlan-domain-name</i>	Associates the interface with the VLAN domain. <code>apic1(config-spine-if)# vlan-domain member evpn-dom</code>
Step 9	exit	Returns to spine configuration mode. <code>apic1(config-spine-if)# exit</code>
Step 10	interface ethernet <i>sub-interface-id</i>	Creates a sub-interface. <code>apic1(config-spine)# interface ethernet 1/33.4</code>
Step 11	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i>	Associates the interface with the overlay-1 VRF and the infra tenant. <code>apic1(config-spine-if)# vrf member tenant infra vrf overlay-1</code>
Step 12	mtu <i>mtu-value</i>	Sets the maximum transmission unit (MTU) for the interface. <code>apic1(config-spine-if)# mtu 1500</code>

	Command or Action	Purpose
Step 13	ip address <i>A.B.C.D/LEN</i>	Sets the IP address for the interface. apicl(config-spine-if) # ip address 5.0.0.1/24
Step 14	ip router ospf default area <i>ospf-area-id</i>	Sets the default OSPF area ID for the interface. apicl(config-spine-if) # ip router ospf default area 0.0.0.150
Step 15	exit	Returns to spine configuration mode. apicl(config-spine-if) # exit
Step 16	interface ethernet <i>slot/port</i>	Configures an interface for a spine node. apicl(config-spine) # interface ethernet 1/34
Step 17	vlan-domain member <i>vlan-domain-name</i>	Associates the interface with the VLAN domain. apicl(config-spine-if) # vlan-domain member evpn-dom
Step 18	exit	Returns to spine configuration mode. apicl(config-spine-if) # exit
Step 19	interface ethernet <i>sub-interface-id</i>	Creates a sub-interface. apicl(config-spine) # interface ethernet 1/34.4
Step 20	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i>	Associates the interface with the overlay-1 VRF and the infra tenant. apicl(config-spine-if) # vrf member tenant infra vrf overlay-1
Step 21	mtu <i>mtu-value</i>	Sets the maximum transmission unit (MTU) for the interface. apicl(config-spine-if) # mtu 1500
Step 22	ip address <i>A.B.C.D/LEN</i>	Sets the IP address for the interface. apicl(config-spine-if) # ip address 2.0.0.1/24
Step 23	ip router ospf default area <i>ospf-area-id</i>	Sets the default OSPF area ID for the interface. apicl(config-spine-if) # ip router ospf default area 0.0.0.200
Step 24	exit	Returns to spine configuration mode. apicl(config-spine-if) # exit
Step 25	router ospf default	Configures OSPF for the spine. apicl(config-spine) # router ospf default
Step 26	vrf member tenant <i>tenant-name</i> vrf <i>vrf-name</i>	Associates the Router OSPF policy with the overlay-1 VRF and infra tenant.

	Command or Action	Purpose
		<code>apic1(config-spine-ospf) # vrf member tenant infra vrf overlay-1</code>
Step 27	<code>area area-id loopback loopback-ip-address</code>	Configure an OSPF area for the OSPF policy. <code>apic1(config-spine-ospf-vrf) # area 0.0.0.150 loopback 10.10.5.3</code>
Step 28	<code>area area-id loopback loopback-ip-address</code>	Configure another OSPF area for the OSPF policy. <code>apic1(config-spine-ospf-vrf) # area 0.0.0.200 loopback 10.10.4.3</code>
Step 29	<code>exit</code>	Returns to spine OSPF configuration mode. <code>apic1(config-spine-ospf-vrf) # exit</code>
Step 30	<code>exit</code>	Returns to spine configuration mode. <code>apic1(config-spine-ospf) # exit</code>

APIC GOLF Connections Shared by Multi-Site Sites

For APIC Sites in a Multi-Site topology, if stretched VRFs share GOLF connections, follow these guidelines to avoid the risk of cross-VRF traffic issues.

Route Target Configuration between the Spine Switches and the DCI

There are two ways to configure EVPN route targets (RTs) for the GOLF VRFs: Manual RT and Auto RT. The route target is synchronized between ACI spines and DCIs through OpFlex. Auto RT for GOLF VRFs has the Fabric ID embedded in the format: – ASN: [FabricID] VNID

If two sites have VRFs deployed as in the following diagram, traffic between the VRFs can be mixed.

Site 1	Site 2
ASN: 100, Fabric ID: 1	ASN: 100, Fabric ID: 1
VRF A: VNID 1000 Import/Export Route Target: 100: [1] 1000	VRF A: VNID 2000 Import/Export Route Target: 100: [1] 2000
VRF B: VNID 2000 Import/Export Route Target: 100: [1] 2000	VRF B: VNID 1000 Import/Export Route Target: 100: [1] 1000

Route Maps Required on the DCI

Since tunnels are not created across sites when transit routes are leaked through the DCI, the churn in the control plane must be reduced as well. EVPN type-5 and type-2 routes sent from GOLF spine in one site towards the DCI should not be sent to GOLF spine in another site. This can happen when the DCI to spine switches have the following types of BGP sessions:

Site1 — IBGP ---- DCI ---- EBGP ---- Site2

Site1 — EBGP ---- DCI ---- IBGP ---- Site2

Site1 — EBGp ---- DCI ---- EBGp ---- Site2

Site1 — IBGP RR client ---- DCI (RR)---- IBGP ---- Site2

To avoid this happening on the DCI, route maps are used with different BGP communities on the inbound and outbound peer policies.

When routes are received from the GOLF spine at one site, the outbound peer policy towards the GOLF spine at another site filters the routes based on the community in the inbound peer policy. A different outbound peer policy strips off the community towards the WAN. All the route-maps are at peer level.

Recommended Shared GOLF Configuration Using the NX-OS Style CLI

Use the following steps to configure route maps and BGP to avoid cross-VRF traffic issues when sharing GOLF connections with a DCI between multiple APIC sites that are managed by Multi-Site.

Step 1 Configure the inbound route map

Example:

```
Inbound peer policy to attach community:
route-map multi-site-in permit 10
set community 1:1 additive
```

Step 2 Configure the outbound peer policy to filter routes based on the community in the inbound peer policy.

Example:

```
ip community-list standard test-com permit 1:1
route-map multi-site-out deny 10
    match community test-com exact-match
route-map multi-site-out permit 11
```

Step 3 Configure the outbound peer policy to filter the community towards the WAN.

Example:

```
ip community-list standard test-com permit 1:1
route-map multi-site-wan-out permit 11
    set comm-list test-com delete
```

Step 4 Configure BGP.

Example:

```
router bgp 1
    address-family l2vpn evpn
    neighbor 11.11.11.11 remote-as 1
    update-source loopback0
    address-family l2vpn evpn
        send-community both
```

```

route-map multi-site-in in
neighbor 13.0.0.2 remote-as 2
address-family l2vpn evpn
send-community both
route-map multi-site-out out

```

Configuring BGP to Support BGP EVPN on a Spine, Using the NX-OS Style CLI

This task shows how to configure BGP on the spine to support BGP EVPN in the following steps:

Procedure

	Command or Action	Purpose
Step 1	configure	Enters global configuration mode. apic1# configure
Step 2	spine <i>spine-name</i>	Creates the spine or enters spine configuration mode. apic1(config)# spine 111
Step 3	router bgp <i>AS-number</i>	Configures BGP for the spine node. apic1(config-spine)# router bgp 100
Step 4	vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i>	Associates the Router BGP policy with the infra tenant and the overlay-1 VRF. apic1(config-spine-bgp)# vrf context tenant infra vrf overlay-1
Step 5	vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i>	Associates the Router BGP policy with the infra tenant and the overlay-1 VRF. apic1(config-spine-bgp-vrf)# vrf context tenant infra vrf overlay-1
Step 6	neighbor neighbor-ip-address evpn	Configures the IP address for an EVPN BGP neighbor. apic1(config-spine-bgp-vrf)# neighbor 10.10.4.1 evpn
Step 7	label <i>label-name</i>	Assigns a label to the neighbor. apic1(config-spine-bgp-vrf-neighbor)# label evpn-aci
Step 8	update-source loopback <i>loopback-ip-address</i> vrf <i>vrf-name</i>	Sets the update source to be the neighbor loopback IP address. apic1(config-spine-bgp-vrf-neighbor)# update-source loopback 10.10.4.3

	Command or Action	Purpose
Step 9	remote-as <i>AS-number</i>	Specifies the autonomous system (AS) number of the neighbor. The valid value can be from 1 to 4294967295. apicl(config-spine-bgp-vrf-neighbor) # remote-as 100
Step 10	exit	Returns to BGP VRF configuration mode. apicl(config-spine-bgp-vrf-neighbor) # exit
Step 11	neighbor neighbor-ip-address evpn	Configures the IP address for an EVPN BGP neighbor. apicl(config-spine-bgp-vrf) # neighbor 10.10.5.1 evpn
Step 12	label <i>label-name</i>	Assigns a label to the neighbor. apicl(config-spine-bgp-vrf-neighbor) # label evpn-aci2
Step 13	update-source loopback <i>loopback-ip-address vrf vrf-name</i>	Sets the update source to be the neighbor loopback IP address. apicl(config-spine-bgp-vrf-neighbor) # update-source loopback 10.10.5.3
Step 14	remote-as <i>AS-number</i>	Specifies the autonomous system (AS) number of the neighbor. The valid value can be from 1 to 4294967295. apicl(config-spine-bgp-vrf-neighbor) # remote-as 100

Configuring a Tenant for BGP EVPN Using the NX-OS Style CLI

This task shows how to configure a tenant for BGP EVPN in the following steps:

Procedure

	Command or Action	Purpose
Step 1	configure	Enters global configuration mode. apicl# configure
Step 2	tenant <i>tenant-name</i>	Creates the tenant or enters tenant configuration mode. apicl(config)# tenant sky
Step 3	vrf context <i>vrf-name</i>	Creates a VRF for the tenant. apicl(config-tenant)# vrf context vrf-sky
Step 4	exit	Returns to tenant configuration mode. apicl(config-tenant-vrf) # exit
Step 5	bridge-domain <i>bd-name</i>	Creates a bridge domain apicl(config-tenant)# bridge-domain bd-sky

	Command or Action	Purpose
Step 6	vrf member <i>vrf-name</i>	Associates the bridge domain with the VRF and tenant. <code>apic1(config-tenant-bd) # vrf member vrf-sky</code>
Step 7	exit	Returns to tenant configuration mode. <code>apic1(config-tenant-bd) # exit</code>
Step 8	interface bridge-domain <i>bd-name</i>	Creates an interface for a bridge domain. <code>apic1(config-tenant) # interface bridge-domain bd_sky</code>
Step 9	ip address <i>A.B.C.D/LEN</i>	Assigns an IP address and length to the bridge-domain interface. <code>apic1(config-tenant-interface) # ip address 59.10.1.1/24</code>
Step 10	exit	Returns to tenant configuration mode. <code>apic1(config-tenant-interface) # exit</code>
Step 11	bridge-domain <i>bd-name</i>	Creates a bridge domain <code>apic1(config-tenant) # bridge-domain bd-sky2</code>
Step 12	vrf member <i>vrf-name</i>	Associates the bridge domain with the VRF and tenant. <code>apic1(config-tenant-bd) # vrf member vrf-sky</code>
Step 13	exit	Returns to tenant configuration mode. <code>apic1(config-tenant-bd) # exit</code>
Step 14	interface bridge-domain <i>bd-name</i>	Creates an interface for a bridge domain. <code>apic1(config-tenant) # interface bridge-domain bd_sky2</code>
Step 15	ip address <i>A.B.C.D/LEN</i>	Assigns an IP address and length to the bridge-domain interface. <code>apic1(config-tenant-interface) # ip address 59.11.1.1/24</code>
Step 16	exit	Returns to tenant configuration mode. <code>apic1(config-tenant-interface) # exit</code>

Configuring a Route Map

This task shows how to configure a route map to advertise bridge-domain subnets through BGP EVPN. Each bridge domain is advertised through a different BGP EVPN session on the spine, with a unique provider label.

Procedure

	Command or Action	Purpose
Step 1	configure	Enters configuration mode. <code>apic1# configure</code>
Step 2	spine <i>spine-name</i>	Creates a spine or enters spine configuration mode. <code>apic1(config)# spine 111</code>
Step 3	vrf context tenant <i>tenant-name</i> vrf <i>vrf-name</i>	Enters creates a VRF or enters VRF configuration mode. <code>apic1(config-spine)# vrf context tenant sky vrf vrf_sky</code>
Step 4	address-family { ipv4 ipv6 } unicast	Sets IPv4 or IPv6 unicast address family for the VRF. <code>apic1(config-spine-vrf)# address-family ipv4 unicast</code>
Step 5	route-target <i>mode</i> extended-community-number	Assigns an export route target to the address family. <code>apic1(config-spine-vrf-af)# route-target export 100:1</code>
Step 6	route-target <i>mode</i> extended-community-number	Assigns an import route target to the address family. <code>apic1(config-spine-vrf-af)# route-target import 100:1</code>
Step 7	exit	Returns to spine VRF configuration mode. <code>apic1(config-spine-vrf-af)# exit</code>
Step 8	route-map <i>route-map-name</i>	Creates a route map for EVPN (with prefix learned from a transit network). <code>apic1(config-spine-vrf)# route map rmap</code>
Step 9	ip prefix-list <i>ip-pl-name</i> permit <i>A.B.C.D/LEN</i>	Adds an IP prefix list to the route map to permit traffic from the specified subnet. <code>apic1(config-spine-vrf-route-map)# ip prefix-list pl permit 11.10.10.0/24</code>
Step 10	match bridge-domain <i>bd-name</i>	Configures the route-map to match traffic belonging to the bridge domain. <code>apic1(config-spine-vrf-route-map)# match bridge-domain bd_sky</code>
Step 11	exit	Returns to spine VRF route-map configuration mode. <code>apic1(config-spine-vrf-route-map-match)# exit</code>
Step 12	match prefix-list <i>pl-name</i>	Sets the route-map to match the specified prefix-list. <code>apic1(config-spine-vrf-route-map)# match prefix-list pl</code>
Step 13	exit	Returns to spine VRF route-map configuration mode.

	Command or Action	Purpose
		<code>apic1(config-spine-vrf-route-map-match) # exit</code>
Step 14	exit	Returns to spine VRF configuration mode. <code>apic1(config-spine-vrf-route-map) # exit</code>
Step 15	evpn export map <i>route-map-name</i> label <i>consumer-label-name</i>	Assigns a consumer label to the VRF. <code>apic1(config-spine-vrf) # evpn export map rmap label evpn-aci</code>
Step 16	route-map <i>route-map-name</i>	Creates a route map for EVPN (with prefix learned from a transit network). <code>apic1(config-spine-vrf) # route map rmap2</code>
Step 17	match bridge-domain <i>bd-name</i>	Configures the route-map to match traffic belonging to the bridge domain. <code>apic1(config-spine-vrf-route-map) # match bridge-domain bd_sky</code>
Step 18	exit	Returns to spine VRF route-map configuration mode. <code>apic1(config-spine-vrf-route-map-match) # exit</code>
Step 19	match prefix-list <i>pl-name</i>	Sets the route-map to match the specified prefix-list. <code>apic1(config-spine-vrf-route-map) # match prefix-list pl</code>
Step 20	exit	Returns to spine VRF route-map configuration mode. <code>apic1(config-spine-vrf-route-map-match) # exit</code>
Step 21	exit	Returns to spine VRF configuration mode. <code>apic1(config-spine-vrf-route-map) # exit</code>
Step 22	evpn export map <i>route-map-name</i> label <i>consumer-label-name</i>	Assigns a consumer label to the VRF. <code>apic1(config-spine-vrf) # evpn export map rmap label evpn-aci2</code>
Step 23	external-l3 <i>epgepg-name</i>	<code>apic1(config-spine-vrf) # external-l3 epg l3_sky</code>
Step 24	vrf member <i>vrf-name</i>	<code>apic1(config-spine-vrf-l3ext-epg) # vrf member vrf_sky</code>
Step 25	match ip <i>A.B.C.D/LEN</i>	Configure the subnet that identifies hosts as being part of the EPG. <code>apic1(config-spine-vrf-l3ext-epg) # match ip 80.10.1.0/24</code>

Enabling Distributing BGP EVPN Type-2 Host Routes to a DCIG Using the NX-OS Style CLI

Procedure

	Command or Action	Purpose
Step 1	Configure distributing EVPN type-2 host routes to a DCIG with the following commands in the BGP address family configuration mode. Example: <pre>apic1(config)# leaf 101 apic1(config-leaf)# template bgp address-family bgpAf1 tenant bgp_t1 apic1(config-bgp-af)# distance 250 240 230 apic1(config-bgp-af)# host-rt-enable apic1(config-bgp-af)# exit</pre>	This template will be available on all nodes where tenant bgp_t1 has a VRF deployment. To disable distributing EVPN type-2 host routes, enter the no host-rt-enable command.

Cisco ACI GOLF Configuration Example, Using the NX-OS Style CLI

These examples show the CLI commands to configure GOLF Services, which uses the BGP EVPN protocol over OSPF for WAN routers that are connected to spine switches.

Configuring the infra Tenant for BGP EVPN

The following example shows how to configure the infra tenant for BGP EVPN, including the VLAN domain, VRF, Interface IP addressing, and OSPF:

```
configure
vlan-domain evpn-dom dynamic
exit
spine 111
    # Configure Tenant Infra VRF overlay-1 on the spine.
    vrf context tenant infra vrf overlay-1
        router-id 10.10.3.3
        exit

interface ethernet 1/33
    vlan-domain member golf_dom
    exit
interface ethernet 1/33.4
    vrf member tenant infra vrf overlay-1
    mtu 1500
    ip address 5.0.0.1/24
    ip router ospf default area 0.0.0.150
    exit
interface ethernet 1/34
    vlan-domain member golf_dom
    exit
interface ethernet 1/34.4
    vrf member tenant infra vrf overlay-1
    mtu 1500
    ip address 2.0.0.1/24
    ip router ospf default area 0.0.0.200
```

```

    exit

router ospf default
  vrf member tenant infra vrf overlay-1
    area 0.0.0.150 loopback 10.10.5.3
    area 0.0.0.200 loopback 10.10.4.3
  exit
exit

```

Configuring BGP on the Spine Node

The following example shows how to configure BGP to support BGP EVPN:

```

Configure
spine 111
router bgp 100
  vrf member tenant infra vrf overlay- 1
    neighbor 10.10.4.1 evpn
    label golf_aci
    update-source loopback 10.10.4.3
    remote-as 100
    exit
  neighbor 10.10.5.1 evpn
  label golf_aci2
  update-source loopback 10.10.5.3
  remote-as 100
  exit
exit
exit

```

Configuring a Tenant for BGP EVPN

The following example shows how to configure a tenant for BGP EVPN, including a gateway subnet which will be advertised through a BGP EVPN session:

```

configure
tenant sky
  vrf context vrf_sky
  exit
  bridge-domain bd_sky
  vrf member vrf_sky
  exit
  interface bridge-domain bd_sky
  ip address 59.10.1.1/24
  exit
  bridge-domain bd_sky2
  vrf member vrf_sky
  exit
  interface bridge-domain bd_sky2
  ip address 59.11.1.1/24
  exit
exit

```

Configuring the BGP EVPN Route Target, Route Map, and Prefix EPG for the Tenant

The following example shows how to configure a route map to advertise bridge-domain subnets through BGP EVPN.

```

configure
spine 111

```

```

vrf context tenant sky vrf vrf_sky
  address-family ipv4 unicast
    route-target export 100:1
    route-target import 100:1
  exit

  route-map rmap
    ip prefix-list p1 permit 11.10.10.0/24
    match bridge-domain bd_sky
    exit
    match prefix-list p1
    exit

  evpn export map rmap label golf_aci

  route-map rmap2
    match bridge-domain bd_sky
    exit
    match prefix-list p1
    exit
  exit

  evpn export map rmap label golf_aci2

external-l3 epg l3_sky
  vrf member vrf_sky
  match ip 80.10.1.0/24
  exit

```

Troubleshooting EVPN Type-2 Route Distribution to a DCIG

For optimal traffic forwarding in an EVPN topology, you can enable fabric spines to distribute host routes to a Data Center Interconnect Gateway (DCIG) using EVPN type-2 (MAC-IP) routes along with the public BD subnets in the form of BGP EVPN type-5 (IP Prefix) routes. This is enabled using the HostLeak object. If you encounter problems with route distribution, use the steps in this topic to troubleshoot.

SUMMARY STEPS

1. Verify that HostLeak object is enabled under the VRF-AF in question, by entering a command such as the following in the spine-switch CLI:
2. Verify that the config-MO has been successfully processed by BGP, by entering a command such as the following in the spine-switch CLI:
3. Verify that the public BD-subnet has been advertised to DCIG as an EVPN type-5 route:
4. Verify whether the host route advertised to the EVPN peer was an EVPN type-2 MAC-IP route:
5. Verify that the EVPN peer (a DCIG) received the correct type-2 MAC-IP route and the host route was successfully imported into the given VRF, by entering a command such as the following on the DCIG device (assuming that the DCIG is a Cisco ASR 9000 switch in the example below):

DETAILED STEPS

Step 1 Verify that HostLeak object is enabled under the VRF-AF in question, by entering a command such as the following in the spine-switch CLI:

Example:

```
spine1# ls /mit/sys/bgp/inst/dom-apple/af-ipv4-ucast/
ctrl-l2vpn-evpn ctrl-vpnv4-ucast hostleak summary
```

Step 2 Verify that the config-MO has been successfully processed by BGP, by entering a command such as the following in the spine-switch CLI:

Example:

```
spine1# show bgp process vrf apple
```

Look for output similar to the following:

```
Information for address family IPv4 Unicast in VRF apple
Table Id           : 0
Table state        : UP
Table refcount     : 3
Peers              : 0
Active-peers       : 0
Routes             : 0
Paths              : 0
Networks           : 0
Aggregates         : 0

Redistribution
  None

Wait for IGP convergence is not configured
GOLF EVPN MAC-IP route is enabled
EVPN network next-hop 192.41.1.1
EVPN network route-map map_pfxleakctrl_v4
Import route-map rtctrlmap-apple-v4
EVPN import route-map rtctrlmap-evpn-apple-v4
```

Step 3 Verify that the public BD-subnet has been advertised to DCIG as an EVPN type-5 route:

Example:

```
spine1# show bgp l2vpn evpn 10.6.0.0 vrf overlay-1
Route Distinguisher: 192.41.1.5:4123 (L3VNI 2097154)
BGP routing table entry for [5]:[0]:[0]:[16]:[10.6.0.0]:[0.0.0.0]/224, version 2088
Paths: (1 available, best #1)
Flags: (0x000002 00000000) on xmit-list, is not in rib/evpn
Multipath: eBGP iBGP

Advertised path-id 1
Path type: local 0x4000008c 0x0 ref 1, path is valid, is best path
AS-Path: NONE, path locally originated
192.41.1.1 (metric 0) from 0.0.0.0 (192.41.1.5)
Origin IGP, MED not set, localpref 100, weight 32768
Received label 2097154
Community: 1234:444
Extcommunity:
  RT:1234:5101
  4BYTEAS-GENERIC:T:1234:444
```

Path-id 1 advertised to peers:
50.41.50.1

In the **Path type** entry, **ref 1** indicates that one route was sent.

Step 4 Verify whether the host route advertised to the EVPN peer was an EVPN type-2 MAC-IP route:

Example:

```
spine1# show bgp l2vpn evpn 10.6.41.1 vrf overlay-1
Route Distinguisher: 10.10.41.2:100 (L2VNI 100)
BGP routing table entry for [2]:[0]:[2097154]:[48]:[0200.0000.0002]:[32]:[10.6.41.1]/272, version 1146
Shared RD: 192.41.1.5:4123 (L3VNI 2097154)
```

```

Paths: (1 available, best #1)
Flags: (0x00010a 00000000) on xmit-list, is not in rib/evpn
Multipath: eBGP iBGP

Advertised path-id 1
Path type: local 0x4000008c 0x0 ref 0, path is valid, is best path
AS-Path: NONE, path locally originated
EVPN network: [5]:[0]:[0]:[16]:[10.6.0.0]:[0.0.0.0] (VRF apple)
  10.10.41.2 (metric 0) from 0.0.0.0 (192.41.1.5)
    Origin IGP, MED not set, localpref 100, weight 32768
    Received label 2097154 2097154
    Extcommunity:
      RT:1234:16777216

```

```

Path-id 1 advertised to peers:
50.41.50.1

```

The **Shared RD** line indicates the RD/VNI shared by the EVPN type-2 route and the BD subnet.

The **EVPN Network** line shows the EVPN type-5 route of the BD-Subnet.

The **Path-id advertised to peers** indicates the path advertised to EVPN peers.

Step 5

Verify that the EVPN peer (a DCIG) received the correct type-2 MAC-IP route and the host route was successfully imported into the given VRF, by entering a command such as the following on the DCIG device (assuming that the DCIG is a Cisco ASR 9000 switch in the example below):

Example:

```

RP/0/RSP0/CPU0:asr9k#show bgp vrf apple-2887482362-8-1 10.6.41.1
Tue Sep  6 23:38:50.034 UTC
BGP routing table entry for 10.6.41.1/32, Route Distinguisher: 44.55.66.77:51
Versions:
  Process          bRIB/RIB  SendTblVer
  Speaker          2088      2088
Last Modified: Feb 21 08:30:36.850 for 28w2d
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    192.41.1.1 (metric 42) from 10.10.41.1 (192.41.1.5)
    Received Label 2097154
    Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported
    Received Path ID 0, Local Path ID 1, version 2088
    Community: 1234:444
    Extended community: 0x0204:1234:444 Encapsulation Type:8 Router
MAC:0200.c029.0101 RT:1234:5101
  RIB RNH: table_id 0xe0000190, Encap 8, VNI 2097154, MAC Address: 0200.c029.0101,
  IP Address: 192.41.1.1, IP table_id 0x00000000
  Source AFI: L2VPN EVPN, Source VRF: default,
  Source Route Distinguisher: 192.41.1.5:4123

```

In this output, the received RD, next hop, and attributes are the same for the type-2 route and the BD subnet.

Multipod_Fabric

About Multipod Fabric

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.

Assigning Switches in a Multipod Fabric

Before you begin

The node group and L3Out policies have already been created.

Procedure

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enter global configuration mode.
Step 2	[no] system switch-id <i>serial-number switch-id switch-name</i> [pod <i>pod-id</i>] [role {leaf spine}] Example: <code>apic1(config)# system switch-id SAL1748H56D 201 ifav4-spine1 pod 1 role spine</code>	For each switch in the multipod fabric, declare the associated pod and the role (leaf or spine) of the switch. Repeat this command for each leaf and spine switch in the multipod fabric.
Step 3	[no] system pod <i>pod-id tep-pool ip-prefix/length</i> Example: <code>apic1(config)# system pod 1 tep-pool 10.0.0.0/16</code>	Configure a tunnel endpoint IP address pool for a pod. Repeat this command for each pod in the multipod fabric.

Example

This example shows how to assign spine and leaf switches in a two-pod fabric.

```
apic1# configure
apic1(config)# system switch-id SAL1748H56D 201 ifav4-spine1 pod 1 role spine
```

```

apic1(config)# system switch-id SAL1938P7A6 202 ifav4-spine3 pod 1 role spine
apic1(config)# system switch-id SAL1819RXP4 101 ifav4-leaf1 pod 1 role leaf
apic1(config)# system switch-id SAL1803L25H 102 ifav4-leaf2 pod 1 role leaf
apic1(config)# system switch-id SAL1934MNY0 103 ifav4-leaf3 pod 1 role leaf
apic1(config)# system switch-id SAL1934MNY3 104 ifav4-leaf4 pod 1 role leaf
apic1(config)# system switch-id SAL1931LA3B 203 ifav4-spine2 pod 2 role spine
apic1(config)# system switch-id FGE173400A9 204 ifav4-spine4 pod 2 role spine
apic1(config)# system switch-id SAL1938PHBB 105 ifav4-leaf5 pod 2 role leaf
apic1(config)# system switch-id SAL1942R857 106 ifav4-leaf6 pod 2 role leaf
apic1(config)# system pod 1 tep-pool 10.0.0.0/16
apic1(config)# system pod 2 tep-pool 10.1.0.0/16

```

What to do next

Configure fabric-external connectivity.

Configuring Fabric-External Connectivity for a Multipod Fabric

Before you begin

- The node group and L3Out policies have already been created.
- Switches have been assigned to pods.

Procedure

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enter global configuration mode.
Step 2	[no] fabric-external <i>controller-number</i> Example: apic1(config)# fabric-external 1	
Step 3	[no] bgp evpn peering [password <i>peering-password</i>] [type {automatic_with_full_mesh automatic_with_rr}] Example: apic1(config-fabric-external)# bgp evpn peering	Configure BGP EVPN peering profile. You can configure a peering password, and you can set the type to be either full mesh or with route-reflector.
Step 4	[no] pod <i>pod-id</i> Example: apic1(config-fabric-external)# pod 1	Select a pod for configuring.
Step 5	[no] interpod data hardware-proxy <i>ip-addr/mask</i> Example: apic1(config-fabric-external-pod)# interpod data hardware-proxy 100.11.1.1/32	Configure the anycast hardware-proxy IP address for each pod for inter-pod traffic.

	Command or Action	Purpose
Step 6	[no] bgp evpn peering [password <i>peering-password</i>] [type {automatic_with_full_mesh automatic_with_rr}] Example: <pre>apic1(config-fabric-external-pod) # bgp evpn peering</pre>	
Step 7	exit Example: <pre>apic1(config-fabric-external-pod) # exit</pre>	Return to BGP EVPN peering profile configuration.
Step 8	Repeat steps 4 through 7 for each pod in the multipod fabric.	
Step 9	[no] route-map interpod-import Example: <pre>apic1(config-fabric-external) # route-map interpod-import</pre>	Configure a route-map that contains subnets on the inter-pod network (IPN) that will be allowed into the fabric through the OSPF protocol.
Step 10	[no] ip prefix-list <i>prefix-list-name</i> [permit <i>ip-address/len</i>] Example: <pre>apic1(config-fabric-external-route-map) # ip prefix-list default permit 0.0.0.0/0</pre>	
Step 11	exit Example: <pre>apic1(config-fabric-external-route-map) # exit</pre>	Return to fabric-external configuration mode.
Step 12	[no] route-target extended <i>ASN4:NN</i> Example: <pre>apic1(config-fabric-external) # route-target extended 5:16</pre>	Route targets are carried as extended community attributes. Enter the community number in the AA4:NN2 format: 1-4294967295: 1-65535.
Step 13	exit	

Example

This example shows how to configure fabric-external connectivity for a multipod fabric.

```
apic1# configure
apic1(config) # fabric-external 1
apic1(config-fabric-external) # bgp evpn peering
apic1(config-fabric-external) # pod 1
apic1(config-fabric-external-pod) # interpod data hardware-proxy 100.11.1.1/32
apic1(config-fabric-external-pod) # bgp evpn peering
apic1(config-fabric-external-pod) # exit
apic1(config-fabric-external) # pod 2
apic1(config-fabric-external-pod) # interpod data hardware-proxy 200.11.1.1/32
apic1(config-fabric-external-pod) # bgp evpn peering
apic1(config-fabric-external-pod) # exit
```



```

apic1(config-fabric-external)# route-map interpod-import
apic1(config-fabric-external-route-map)# ip prefix-list default permit 0.0.0.0/0
apic1(config-fabric-external-route-map)# exit
apic1(config-fabric-external)# route-target extended 5:16
apic1(config-fabric-external)# exit

```

What to do next

Configure spine interfaces and OSPF.

Configuring Spine Interfaces and OSPF for a Multipod Fabric

Before you begin

- Switches have been assigned to pods.
- A VLAN domain must exist.

Procedure

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enter global configuration mode.
Step 2	spine <i>spine-id</i> Example: apic1(config)# spine 104	You can specify the spine switch by an ID number in the range of 101 to 4000 or by name, such as 'spine1.'
Step 3	[no] vrf context <i>tenant infra vrf vrf-name</i> Example: apic1(config-spine)# vrf context <i>tenant infra vrf overlay-1</i>	
Step 4	[no] router-id <i>A.B.C.D</i> Example: apic1(config-spine-vrf)# router-id 201.201.201.201	Configure a router identifier (ID).
Step 5	exit Example: apic1(config-spine-vrf)# exit	Return to spine configuration mode.
Step 6	[no] interface <i>ethernet slot/port</i> Example: apic1(config-spine)# interface <i>ethernet 1/1</i>	

	Command or Action	Purpose
Step 7	[no] vlan-domain member <i>domain-name</i> Example: <code>apic1(config-spine)# vlan-domain member 13Dom</code>	The VLAN domain must already exist, having been created using the vlan-domain <i>domain-name</i> command in the global configuration mode.
Step 8	exit Example: <code>apic1(config-spine-if)# exit</code>	Return to spine configuration mode.
Step 9	[no] interface ethernet <i>type/slot.subinterface</i> Example: <code>apic1(config-spine)# interface ethernet 1/1.4</code>	Encapsulation for the subinterface must be 4.
Step 10	[no] vrf member tenant infra vrf <i>vrf-name</i> Example: <code>apic1(config-spine-if)# vrf member tenant infra vrf overlay-1</code>	Configure the interface as a member of the tenant VRF.
Step 11	[no] ip address <i>ip-address</i> Example: <code>apic1(config-spine-if)# ip address 201.1.1.1/30</code>	
Step 12	[no] ip router ospf default area <i>0.0.0.0</i> Example: <code>apic1(config-spine-if)# ip router ospf default area 0.0.0.0</code>	
Step 13	[no] ip ospf cost <i>cost</i> Example: <code>apic1(config-spine-if)# ip ospf cost 1</code>	
Step 14	exit Example: <code>apic1(config-spine-if)# exit</code>	Return to spine configuration mode.
Step 15	Repeat steps Step 6, on page 325 through Step 14, on page 326 to add any additional interfaces.	
Step 16	[no] router ospf default Example: <code>apic1(config-spine)# router ospf default</code>	
Step 17	[no] vrf member tenant infra vrf <i>vrf-name</i> Example: <code>apic1(config-spine-ospf)# vrf member tenant infra vrf overlay-1</code>	

	Command or Action	Purpose
Step 18	[no] area <i>area</i> loopback <i>ip-address</i> Example: <pre>apic1(config-spine-ospf-vrf)# area 0.0.0.0 loopback 201.201.201.201</pre>	Advertise the loopback address through OSPF. This address is used by BGP EVPN sessions for peering.
Step 19	[no] area <i>area</i> interpod peering Example: <pre>apic1(config-spine-ospf-vrf)# area 0.0.0.0 interpod peering</pre>	Enable inter-pod peering on the OSPF area, which will set up BGP EVPN sessions automatically using the loopback address advertised by OSPF.
Step 20	exit Example: <pre>apic1(config-spine-ospf-vrf)# exit</pre>	Return to OSPF configuration mode.
Step 21	exit Example: <pre>apic1(config-spine-ospf)# exit</pre>	Return to spine configuration mode.
Step 22	exit Example: <pre>apic1(config-spine)# exit</pre>	Return to global configuration mode.
Step 23	Repeat steps Step 2, on page 325 through Step 22, on page 327 to configure additional spine switches.	

Example

```
apic1# configure

# CONFIGURE FIRST SPINE

apic1(config)# spine 201
apic1(config-spine)# vrf context tenant infra vrf overlay-1
apic1(config-spine-vrf)# router-id 201.201.201.201
apic1(config-spine-vrf)# exit

apic1(config-spine)# interface ethernet 1/1
apic1(config-spine-if)# vlan-domain member 13Dom
apic1(config-spine-if)# exit
apic1(config-spine)# interface ethernet 1/1.4
apic1(config-spine-if)# vrf member tenant infra vrf overlay-1
apic1(config-spine-if)# ip address 201.1.1.1/30
apic1(config-spine-if)# ip router ospf default area 0.0.0.0
apic1(config-spine-if)# ip ospf cost 1
apic1(config-spine-if)# exit

apic1(config-spine)# interface ethernet 1/2
apic1(config-spine-if)# vlan-domain member 13Dom
apic1(config-spine-if)# exit
apic1(config-spine)# interface ethernet 1/2.4
```

```

apicl(config-spine-if)# vrf member tenant infra vrf overlay-1
apicl(config-spine-if)# ip address 201.2.1.1/30
apicl(config-spine-if)# ip router ospf default area 0.0.0.0
apicl(config-spine-if)# ip ospf cost 1
apicl(config-spine-if)# exit

apicl(config-spine)# router ospf default
apicl(config-spine-ospf)# vrf member tenant infra vrf overlay-1
apicl(config-spine-ospf-vrf)# area 0.0.0.0 loopback 201.201.201.201
apicl(config-spine-ospf-vrf)# area 0.0.0.0 interpod peering
apicl(config-spine-ospf-vrf)# exit
apicl(config-spine-ospf)# exit
apicl(config-spine)# exit

# CONFIGURE SECOND SPINE

apicl(config)# spine 202
apicl(config-spine)# vrf context tenant infra vrf overlay-1
apicl(config-spine-vrf)# router-id 202.202.202.202
apicl(config-spine-vrf)# exit

apicl(config-spine)# interface ethernet 1/2
apicl(config-spine-if)# vlan-domain member l3Dom
apicl(config-spine-if)# exit
apicl(config-spine)# interface ethernet 1/2.4
apicl(config-spine-if)# vrf member tenant infra vrf overlay-1
apicl(config-spine-if)# ip address 202.1.1.1/30
apicl(config-spine-if)# ip router ospf default area 0.0.0.0
apicl(config-spine-if)# ip ospf cost 1
apicl(config-spine-if)# exit

apicl(config-spine)# router ospf default
apicl(config-spine-ospf)# vrf member tenant infra vrf overlay-1
apicl(config-spine-ospf-vrf)# area 0.0.0.0 loopback 202.202.202.202
apicl(config-spine-ospf-vrf)# area 0.0.0.0 interpod peering
apicl(config-spine-ospf-vrf)# exit
apicl(config-spine-ospf)# exit
apicl(config-spine)# exit

# CONFIGURE ADDITIONAL SPINES

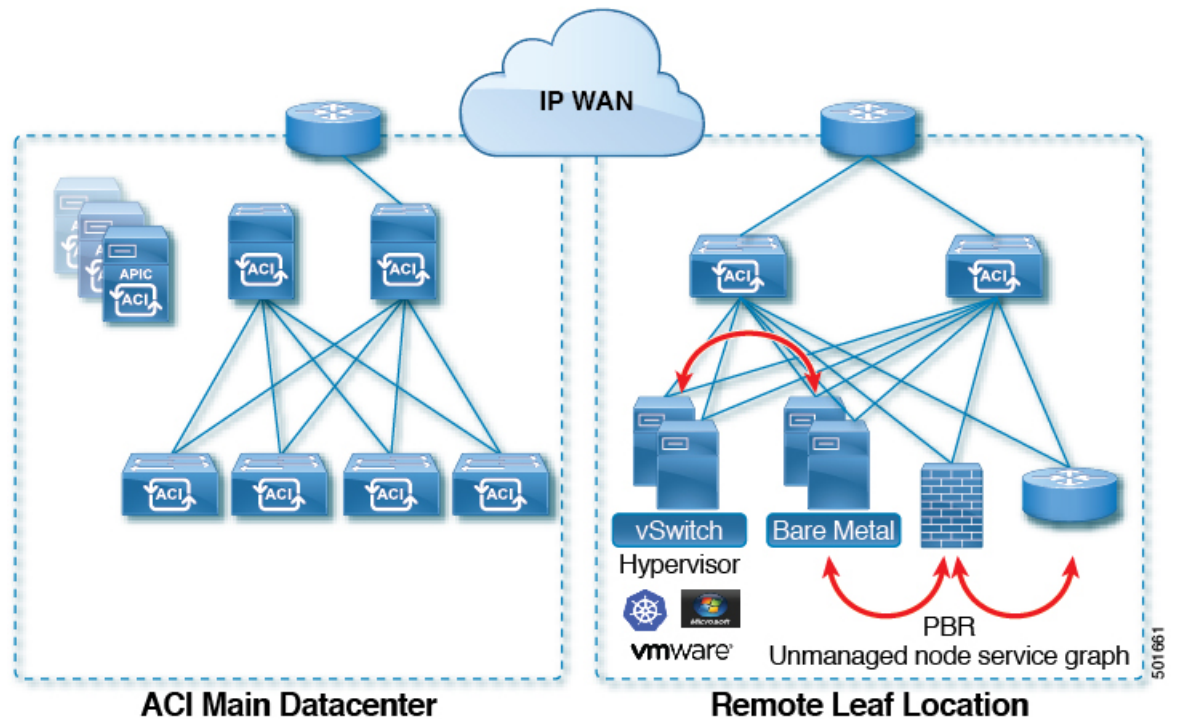
```

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 datacenters with Cisco ACI leaf switches that have no local spine switch or APIC attached.

Figure 24: Remote Leaf Topology



The remote leaf switches are added to an existing pod in the fabric. All policies deployed in the main datacenter 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 Multicast. All local traffic on the remote site is switched directly between endpoints, whether physical or virtual. Any traffic that requires use of the spine switch proxy is forwarded to the main datacenter.

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.
- Before decommissioning a remote leaf, you must first delete the vPC.

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 via 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 site when spine-proxy is not reachable.

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-C9364C and N9K-C9332C
- Modular spine switches with N9K-X9732C-EX or N9K-X9736C-FX linecards
- 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.

Restrictions and Limitations



Note In Cisco APIC Release 4.0(x), the following features are supported that were not previously:

- 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

Stretching of L3out SVI between local leaf switches (ACI main data center switches) and remote leaf switches is not supported.

The following deployments and configurations are not supported with the remote leaf switch feature:

- Spanning Tree Protocol across remote leaf site and main data center
- APIC controllers 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

Full fabric and tenant policies are supported on remote leaf switches, in this release, except for the following features:

- ACI Multi-Site
- 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
- PTP
- Layer 3 Multicast routing on remote leaf switches
- Maintenance mode
- TEP to TEP atomic counters
- Transit L3Out across remote locations, which is when the main Cisco ACI datacenter 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)
- Traffic forwarding directly across two remote leaf vPC pairs in the same remote datacenter or across datacenters

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 NX-OS Style CLI

This example configures a spine switch and a remote leaf switch to enable the leaf switch to communicate with the main fabric pod.

Before you begin

- The IPN router and remote leaf switches are active and configured; see [WAN Router and Remote Leaf Switch Configuration Guidelines, on page 332](#).
- The remote leaf switches are running a switch image of 13.1.x or later (aci-n9000-dk9.13.1.x.x.bin).
- The pod in which you plan to add the remote leaf switches is created and configured.

Step 1 Define the TEP pool for a remote location 5, in pod 2.

The network mask must be /24 or lower.

Use the following new command: **system remote-leaf-site *site-id* pod *pod-id* tep-pool *ip-address-and-netmask***

Example:

```
apic1(config)# system remote-leaf-site 5 pod 2 tep-pool 192.0.0.0/16
```

Step 2 Add a remote leaf switch to pod 2, remote-leaf-site 5.

Use the following command: **system switch-id *serial-number node-id* leaf-switch-name pod *pod-id* remote-leaf-site *remote-leaf-site-id* node-type remote-leaf-wan**

Example:

```
apic1(config)# system switch-id FDO210805SKD 109 ifav4-leaf9 pod 2
remote-leaf-site 5 node-type remote-leaf-wan
```

Step 3 Configure a VLAN domain with a VLAN that includes VLAN 4.

Example:

```
apic1(config)# vlan-domain ospfDom
apic1(config-vlan)# vlan 4-5
apic1(config-vlan)# exit
```

Step 4 Configure two L3Outs for the infra tenant, one for the remote leaf connections and one for the multipod IPN.

Example:

```
apic1(config)# tenant infra
apic1(config-tenant)# l3out rl-wan
apic1(config-tenant-l3out)# vrf member overlay-1
apic1(config-tenant-l3out)# exit
apic1(config-tenant)# l3out ipn-multipodInternal
apic1(config-tenant-l3out)# vrf member overlay-1
apic1(config-tenant-l3out)# exit
apic1(config-tenant)# exit
apic1(config)#
```

Step 5 Configure the spine switch interfaces and sub-interfaces to be used by the L3Outs.

Example:

```
apic1(config)# spine 201
apic1(config-spine)# vrf context tenant infra vrf overlay-1 l3out rl-wan-test
apic1(config-spine-vrf)# exit
apic1(config-spine)# vrf context tenant infra vrf overlay-1 l3out ipn-multipodInternal
apic1(config-spine-vrf)# exit
apic1(config-spine)#
apic1(config-spine)# interface ethernet 8/36
apic1(config-spine-if)# vlan-domain member ospfDom
apic1(config-spine-if)# exit
apic1(config-spine)# router ospf default
apic1(config-spine-ospf)# vrf member tenant infra vrf overlay-1
apic1(config-spine-ospf-vrf)# area 5 l3out rl-wan-test
apic1(config-spine-ospf-vrf)# exit
apic1(config-spine-ospf)# exit
apic1(config-spine)#
apic1(config-spine)# interface ethernet 8/36.4
apic1(config-spine-if)# vrf member tenant infra vrf overlay-1 l3out rl-wan-test
apic1(config-spine-if)# ip router ospf default area 5
apic1(config-spine-if)# exit
apic1(config-spine)# router ospf multipod-internal
apic1(config-spine-ospf)# vrf member tenant infra vrf overlay-1
apic1(config-spine-ospf-vrf)# area 5 l3out ipn-multipodInternal
apic1(config-spine-ospf-vrf)# exit
apic1(config-spine-ospf)# exit
apic1(config-spine)#
apic1(config-spine)# interface ethernet 8/36.5
apic1(config-spine-if)# vrf member tenant infra vrf overlay-1 l3out ipn-multipodInternal
apic1(config-spine-if)# ip router ospf multipod-internal area 5
apic1(config-spine-if)# exit
apic1(config-spine)# exit
apic1(config)#
```

Step 6 Configure the remote leaf switch interface and sub-interface used for communicating with the main fabric pod.

Example:

```
(config)# leaf 101
apic1(config-leaf)# vrf context tenant infra vrf overlay-1 l3out rl-wan-test
apic1(config-leaf-vrf)# exit
apic1(config-leaf)#
apic1(config-leaf)# interface ethernet 1/49
apic1(config-leaf-if)# vlan-domain member ospfDom
apic1(config-leaf-if)# exit
apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant infra vrf overlay-1
apic1(config-leaf-ospf-vrf)# area 5 l3out rl-wan-test
```

```
apic1(config-leaf-ospf-vrf)# exit
apic1(config-leaf-ospf)# exit
apic1(config-leaf)#
apic1(config-leaf)# interface ethernet 1/49.4
apic1(config-leaf-if)# vrf member tenant infra vrf overlay-1 l3out rl-wan-test
apic1(config-leaf-if)# ip router ospf default area 5
apic1(config-leaf-if)# exit
```

Example

The following example provides a downloadable configuration:

```
apic1# configure
apic1(config)# system remote-leaf-site 5 pod 2 tep-pool 192.0.0.0/16
apic1(config)# system switch-id FDO210805SKD 109 ifav4-leaf9 pod 2
remote-leaf-site 5 node-type remote-leaf-wan
apic1(config)# vlan-domain ospfDom
apic1(config-vlan)# vlan 4-5
apic1(config-vlan)# exit
apic1(config)# tenant infra
apic1(config-tenant)# l3out rl-wan-test
apic1(config-tenant-l3out)# vrf member overlay-1
apic1(config-tenant-l3out)# exit
apic1(config-tenant)# l3out ipn-multipodInternal
apic1(config-tenant-l3out)# vrf member overlay-1
apic1(config-tenant-l3out)# exit
apic1(config-tenant)# exit
apic1(config)#
apic1(config)# spine 201
apic1(config-spine)# vrf context tenant infra vrf overlay-1 l3out rl-wan-test
apic1(config-spine-vrf)# exit
apic1(config-spine)# vrf context tenant infra vrf overlay-1 l3out ipn-multipodInternal
apic1(config-spine-vrf)# exit
apic1(config-spine)#
apic1(config-spine)# interface ethernet 8/36
apic1(config-spine-if)# vlan-domain member ospfDom
apic1(config-spine-if)# exit
apic1(config-spine)# router ospf default
apic1(config-spine-ospf)# vrf member tenant infra vrf overlay-1
apic1(config-spine-ospf-vrf)# area 5 l3out rl-wan-test
apic1(config-spine-ospf-vrf)# exit
apic1(config-spine-ospf)# exit
apic1(config-spine)#
apic1(config-spine)# interface ethernet 8/36.4
apic1(config-spine-if)# vrf member tenant infra vrf overlay-1 l3out rl-wan-test
apic1(config-spine-if)# ip router ospf default area 5
apic1(config-spine-if)# exit
apic1(config-spine)# router ospf multipod-internal
apic1(config-spine-ospf)# vrf member tenant infra vrf overlay-1
apic1(config-spine-ospf-vrf)# area 5 l3out ipn-multipodInternal
apic1(config-spine-ospf-vrf)# exit
apic1(config-spine-ospf)# exit
apic1(config-spine)#
apic1(config-spine)# interface ethernet 8/36.5
apic1(config-spine-if)# vrf member tenant infra vrf overlay-1 l3out ipn-multipodInternal
apic1(config-spine-if)# ip router ospf multipod-internal area 5
apic1(config-spine-if)# exit
apic1(config-spine)# exit
apic1(config)#
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant infra vrf overlay-1 l3out rl-wan-test
```

```

apic1(config-leaf-vrf)# exit
apic1(config-leaf)#
apic1(config-leaf)# interface ethernet 1/49
apic1(config-leaf-if)# vlan-domain member ospfDom
apic1(config-leaf-if)# exit
apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant infra vrf overlay-1
apic1(config-leaf-ospf-vrf)# area 5 l3out rl-wan-test
apic1(config-leaf-ospf-vrf)# exit
apic1(config-leaf-ospf)# exit
apic1(config-leaf)#
apic1(config-leaf)# interface ethernet 1/49.4
apic1(config-leaf-if)# vrf member tenant infra vrf overlay-1 l3out rl-wan-test
apic1(config-leaf-if)# ip router ospf default area 5
apic1(config-leaf-if)# exit

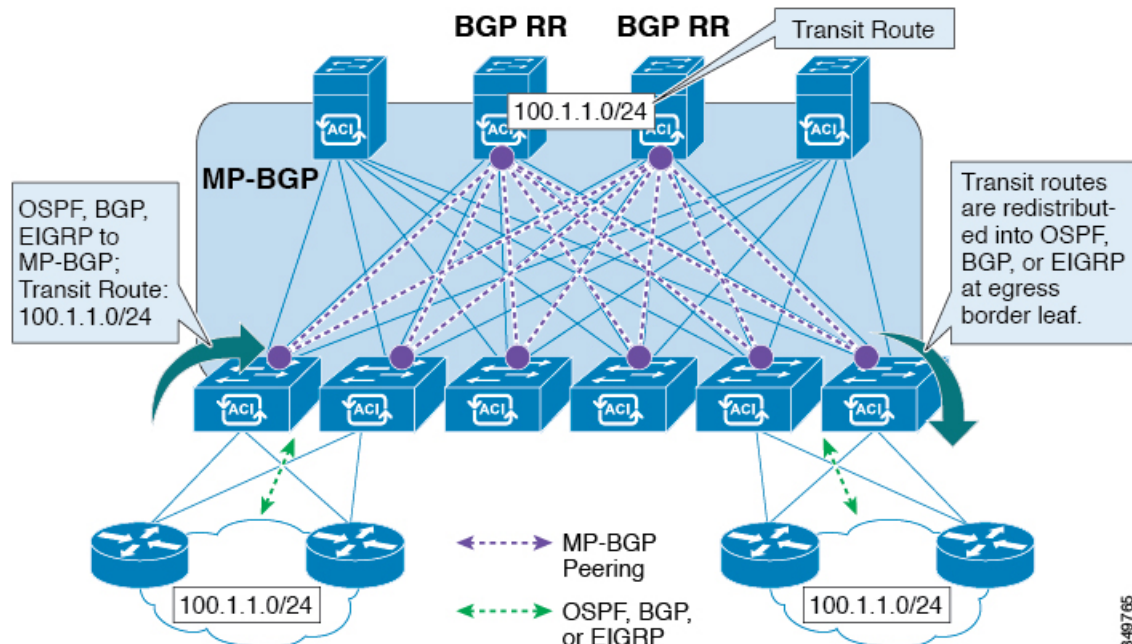
```

Transit Routing

Transit Routing in the ACI Fabric

The Cisco APIC software supports external Layer 3 connectivity with OSPF (NSSA) and iBGP. The fabric advertises the tenant bridge domain subnets out to the external routers on the External Layer 3 Outside (L3Out) connections. The routes that are learned from the external routers are not advertised to other external routers. The fabric behaves like a stub network that can be used to carry the traffic between the external Layer 3 domains.

Figure 25: Transit Routing in the Fabric



In transit routing, multiple L3Out connections within a single tenant and VRF are supported and the APIC advertises the routes that are learned from one L3Out connection to another L3Out connection. The external

Layer 3 domains peer with the fabric on the border leaf switches. The fabric is a transit Multiprotocol-Border Gateway Protocol (MP-BGP) domain between the peers.

The configuration for external L3Out connections is done at the tenant and VRF level. The routes that are learned from the external peers are imported into MP-BGP at the ingress leaf per VRF. The prefixes that are learned from the L3Out connections are exported to the leaf switches only where the tenant VRF is present.



Note For cautions and guidelines for configuring transit routing, see [Guidelines for Transit Routing, on page 338](#)

Transit Routing Related Topics

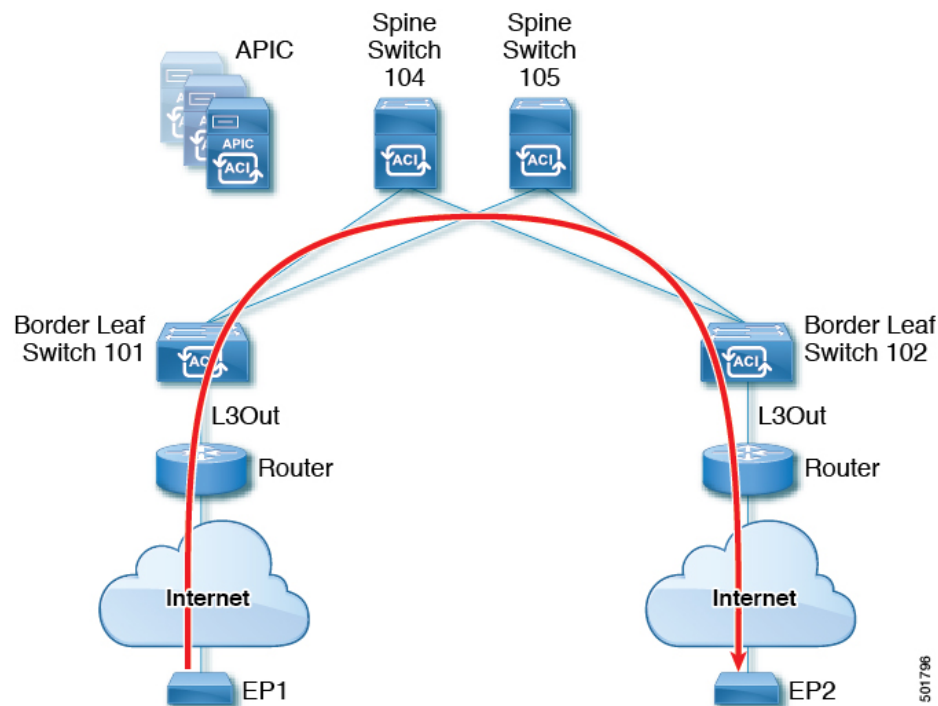
For a transit routing overview, use cases, guidelines, and information about transit routing route controls, see *Cisco APIC Layer 3 Networking Configuration Guide*.

Transit Routing Overview

This topic provides a typical example of how to configure Transit Routing when using Cisco APIC.

The examples in this chapter use the following topology:

Figure 26:



In the examples in this chapter, the Cisco ACI fabric has 2 leaf switches and two spine switches, that are controlled by an APIC cluster. The border leaf switches 101 and 102 have L3Outs on them providing connections to two routers and thus to the Internet. The goal of this example is to enable traffic to flow from EP 1 to EP 2 on the Internet into and out of the fabric through the two L3Outs.

In this example, the tenant that is associated with both L3Outs is `t1`, with VRF `v1`.

Before configuring the L3Outs, configure the nodes, ports, functional profiles, AEPs, and a Layer 3 domain. You must also configure the spine switches 104 and 105 as BGP route reflectors.

Configuring transit routing includes defining the following components:

1. Tenant and VRF
2. Node and interface on leaf 101 and leaf 102
3. Primary routing protocol on each L3Out (used to exchange routes between border leaf switch and external routers; in this example, BGP)
4. Connectivity routing protocol on each L3Out (provides reachability information for the primary protocol; in this example, OSPF)
5. Two external EPGs
6. One route map
7. At least one filter and one contract
8. Associate the contract with the external EPGs


Note

For transit routing cautions and guidelines, see [Guidelines for Transit Routing, on page 338](#).

The following table lists the names that are used in the examples in this chapter:

Property	Names for L3Out1 on Node 101	Names for L3Out2 on Node 102
Tenant	t1	t1
VRF	v1	v1
Node	nodep1 with router ID 11.11.11.103	nodep2 with router ID 22.22.22.203
OSPF Interface	ifp1 at eth/1/3	ifp2 at eth/1/3
BGP peer address	15.15.15.2/24	25.25.25.2/24
External EPG	extnw1 at 192.168.1.0/24	extnw2 at 192.168.2.0/24
Route map	rp1 with ctx1 and route destination 192.168.1.0/24	rp2 with ctx2 and route destination 192.168.2.0/24
Filter	http-filter	http-filter
Contract	httpCtret provided by extnw1	httpCtret consumed by extnw2

Guidelines for Transit Routing

Use the following guidelines when creating and maintaining transit routing connections:

Topic	Caution or Guideline
Transit Routing with a Single L3Out Profile	Before APIC, release 2.3(1f), transit routing was not supported within a single L3Out profile. In APIC, release 2.3(1f) and later, you can configure transit routing with a single L3Out profile, with the following limitations: • If the VRF is unenforced, an external subnet (l3extSubnet) of 0.0.0.0/0 can be used to allow traffic between the routers sharing the same L3EPG. • If the VRF is enforced, an external default subnet (0.0.0.0/0) cannot be used to match both source and destination prefixes for traffic within the same Layer 3 EPG. To match all traffic within the same Layer 3 EPG, the following prefixes are supported: • IPv4 • 0.0.0.0/1—with External Subnets for the External EPG • 128.0.0.0/1—with External Subnets for the External EPG • 0.0.0.0/0—with Import Route Control Subnet, Aggregate Import • IPv6 • 0::0/1—with External Subnets for the External EPG • 8000::0/1—with External Subnets for the External EPG • 0:0/0—with Import Route Control Subnet, Aggregate Import • Alternatively, a single default subnet (0.0.0.0/0) can be used when combined with a VzAny contract. For example: • Use a VzAny provided contract and an Layer 3 EPG consumed contract (matching 0.0.0.0/0), or a VzAny consumed contract and Layer 3 EPG provided contract (matching 0.0.0.0/0). • Use the subnet 0.0.0.0/0—with Import/Export Route Control Subnet, Aggregate Import, and Aggregate Export.

Topic	Caution or Guideline
Shared Routes: Differences in Hardware Support	Routes shared between VRFs function correctly on Generation 2 switches (Cisco Nexus N9K switches with "EX" or "FX" on the end of the switch model name, or later; for example, N9K-93108TC-EX). On Generation 1 switches, however, there may be dropped packets with this configuration, because the physical ternary content-addressable memory (TCAM) tables that store routes do not have enough capacity to fully support route parsing.
OSPF or EIGRP in Back to Back Configuration	Cisco APIC supports transit routing in export route control policies that are configured on the L3Out. These policies control which transit routes (prefixes) are redistributed into the routing protocols in the L3Out. When these transit routes are redistributed into OSPF or EIGRP, they are tagged 4294967295 to prevent routing loops. The Cisco ACI fabric does not accept routes matching this tag when learned on an OSPF or EIGRP L3Out. However, in the following cases, it is necessary to override this behavior: • When connecting two Cisco ACI fabrics using OSPF or EIGRP. • When connecting two different VRFs in the same Cisco ACI fabric using OSPF or EIGRP. Where an override is required, you must configure the VRF with a different tag policy at the following APIC GUI location: Tenant > Tenant_name > Networking > Protocol Policies > Route Tag. Apply a different tag. In addition to creating the new route-tag policy, update the VRF to use this policy at the following APIC GUI location: Tenant > Tenant_name > Networking > VRFs > Tenant_VRF. Apply the route tag policy that you created to the VRF. Note When multiple L3Outs or multiple interfaces in the same L3Out are deployed on the same leaf switch and used for transit routing, the routes are advertised within the IGP (not redistributed into the IGP). In this case the route-tag policy does not apply.

Topic	Caution or Guideline
Advertising BD Subnets Outside the Fabric	The import and export route control policies only apply to the transit routes (the routes that are learned from other external peers) and the static routes. The subnets internal to the fabric that are configured on the tenant BD subnets are not advertised out using the export policy subnets. The tenant subnets are still permitted using the IP prefix-lists and the route-maps but they are implemented using different configuration steps. See the following configuration steps to advertise the tenant subnets outside the fabric: 1. Configure the tenant subnet scope as Public Subnet in the subnet properties window. 2. Optional. Set the Subnet Control as ND RA Prefix in the subnet properties window. 3. Associate the tenant bridge domain (BD) with the external Layer 3 Outside (L3Out). 4. Create contract (provider or consumer) association between the tenant EPG and the external EPG. Setting the BD subnet to Public scope and associating the BD to the L3Out creates an IP prefix-list and the route-map sequence entry on the border leaf for the BD subnet prefix.

Topic	Caution or Guideline
Advertising a Default Route	For external connections to the fabric that only require a default route, there is support for originating a default route for OSPF, EIGRP, and BGP L3Out connections. If a default route is received from an external peer, this route can be redistributed out to another peer following the transit export route control as described earlier in this article. A default route can also be advertised out using a Default Route Leak policy. This policy supports advertising a default route if it is present in the routing table or it always supports advertising a default route. The Default Route Leak policy is configured in the L3Out connection. When creating a Default Route Leak policy, follow these guidelines: • For BGP, the Always property is not applicable. • For BGP, when configuring the Scope property, choose Outside. • For OSPF, the scope value Context creates a type-5 LSA while the Scope value Outside creates type-7 LSA. Your choice depends on the area type configured in the L3Out. If the area type is Regular, set the scope to Context. If the area type is NSSA, set the scope to Outside. • For EIGRP, when choosing the Scope property, you must choose Context.
MTU	Cisco ACI does not support IP fragmentation. Therefore, when you configure Layer 3 Outside (L3Out) connections to external routers, or multipod connections through an Inter-Pod Network (IPN), it is critical that the MTU is set appropriately on both sides. On some platforms, such as ACI, Cisco NX-OS, and Cisco IOS, the configurable MTU value takes into account the IP headers (resulting in a max packet size to be set as 9216 bytes for ACI and 9000 for NX-OS and IOS). However, other platforms such as IOS-XR configure the MTU value exclusive of packet headers (resulting in a max packet size of 8986 bytes). For the appropriate MTU values for each platform, see the relevant configuration guides. Cisco highly recommends you test the MTU using CLI-based commands. For example, on the Cisco NX-OS CLI, use a command such as <code>ping 1.1.1.1 df-bit packet-size 9000 source-interface ethernet 1/1</code>.

Configure Transit Routing Using the NX-OS Style CLI

These steps describe how to configure transit routing for a tenant. This example deploys two L3Outs, in one VRF, on two border leaf switches, that are each connected to separate routers.

Before you begin

- Configure the node, port, functional profile, AEP, and Layer 3 domain.
- Configure a VLAN domain using the **vlan-domain** *domain* and **vlan** *vlan-range* commands.
- Configure a BGP route reflector policy to propagate the routed within the fabric.

For an example of the commands for these prerequisites, see [NX-OS Style CLI Example: L3Out Prerequisites](#), on page 176.

Step 1 Configure the tenant and VRF.

This example configures tenant `t1` with VRF `v1`. The VRF is not yet deployed.

Example:

```
apic1# configure
apic1(config)# tenant t1
apic1(config-tenant)# vrf context v1
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# exit
```

Step 2 Configure the nodes and interfaces.

This example configures two L3Outs for the tenant `t1`, on two border leaf switches:

- The first L3Out is on node 101, which is named `nodep1`. Node 101 is configured with router ID `11.11.11.103`. It has a routed interface `ifp1` at `eth1/3`, with the IP address `12.12.12.3/24`.
- The second L3Out is on node 102, which is named `nodep2`. Node 102 is configured with router ID `22.22.22.203`. It has a routed interface `ifp2` at `eth1/3`, with the IP address, `23.23.23.1/24`.

Example:

```
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# router-id 11.11.11.103
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# interface ethernet 1/3
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member tenant t1 vrf v1
apic1(config-leaf-if)# ip address 12.12.12.3/24
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 102
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# router-id 22.22.22.203
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# interface ethernet 1/3
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member tenant t1 vrf v1
apic1(config-leaf-if)# ip address 23.23.23.3/24
```

```
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

Step 3 Configure the routing protocol for both leaf switches.

This example configures BGP as the primary routing protocol for both the border leaf switches, both with ASN 100. It also configures Node 101 with BGP peer 15.15.15.2 and node 102 with BGP peer 25.25.25.2.

Example:

```
apic1(config)# leaf 101
apic1(config-leaf)# router bgp 100
apic1(config-leaf-bgp)# vrf member tenant t1 vrf v1
apic1(config-leaf-bgp-vrf)# neighbor 15.15.15.2
apic1(config-leaf-bgp-vrf-neighbor)# exit
apic1(config-leaf-bgp-vrf)# exit
apic1(config-leaf-bgp)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 102
apic1(config-leaf)# router bgp 100
apic1(config-leaf-bgp)# vrf member tenant t1 vrf v1
apic1(config-leaf-bgp-vrf)# neighbor 25.25.25.2
apic1(config-leaf-bgp-vrf-neighbor)# exit
apic1(config-leaf-bgp-vrf)# exit
apic1(config-leaf-bgp)# exit
apic1(config-leaf)# exit
```

Step 4 Configure a connectivity routing protocol.

This example configures OSPF as the communication protocol, for both L3Outs, with regular area ID 0.0.0.0.

Example:

```
apic1(config)# leaf 101
apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant t1 vrf v1
apic1(config-leaf-ospf-vrf)# area 0.0.0.0 loopback 40.40.40.1
apic1(config-leaf-ospf-vrf)# exit
apic1(config-leaf-ospf)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 102
apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant t1 vrf v1
apic1(config-leaf-ospf-vrf)# area 0.0.0.0 loopback 60.60.60.1
apic1(config-leaf-ospf-vrf)# exit
apic1(config-leaf-ospf)# exit
apic1(config-leaf)# exit
```

Step 5 Configure the external EPGs.

This example configures the network 192.168.1.0/24 as external network `extnw1` on node 101 and the network 192.168.2.0/24 as external network `extnw2` on node 102.

Example:

```
apic1(config)# tenant t1
apic1(config-tenant)# external-l3 epg extnw1
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# match ip 192.168.1.0/24
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# external-l3 epg extnw2
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# match ip 192.168.2.0/24
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# exit
```

```

apicl(config)# leaf 101
apicl(config-leaf)# vrf context tenant t1 vrf v1
apicl(config-leaf-vrf)# external-l3 epg extnw1
apicl(config-leaf-vrf)# exit
apicl(config-leaf)# exit
apicl(config)# leaf 102
apicl(config-leaf)# vrf context tenant t1 vrf v1
apicl(config-leaf-vrf)# external-l3 epg extnw2
apicl(config-leaf-vrf)# exit
apicl(config-leaf)# exit

```

Step 6 Optional. Configure the route maps.

This example configures a route map for each BGP peer in the inbound and outbound directions.

Example:

Example:

```

apicl(config)# leaf 101
apicl(config-leaf)# template route group match-rule1 tenant t1
apicl(config-route-group)# ip prefix permit 192.168.1.0/24
apicl(config-route-group)# exit
apicl(config-leaf)# template route group match-rule2 tenant t1
apicl(config-route-group)# ip prefix permit 192.168.2.0/24
apicl(config-route-group)# exit
apicl(config-leaf)# vrf context tenant t1 vrf v1
apicl(config-leaf-vrf)# route-map rp1
apicl(config-leaf-vrf-route-map)# match route group match-rule1 order 0
apicl(config-leaf-vrf-route-map-match)# exit
apicl(config-leaf-vrf-route-map)# exit
apicl(config-leaf-vrf)# route-map rp2
apicl(config-leaf-vrf-route-map)# match route group match-rule2 order 0
apicl(config-leaf-vrf-route-map-match)# exit
apicl(config-leaf-vrf-route-map)# exit
apicl(config-leaf-vrf)# exit
apicl(config-leaf)# router bgp 100
apicl(config-leaf-bgp)# vrf member tenant t1 vrf v1
apicl(config-leaf-bgp-vrf)# neighbor 15.15.15.2
apicl(config-leaf-bgp-vrf-neighbor)# route-map rp1 in
apicl(config-leaf-bgp-vrf-neighbor)# route-map rp2 out
apicl(config-leaf-bgp-vrf-neighbor)# exit
apicl(config-leaf-bgp-vrf)# exit
apicl(config-leaf-bgp)# exit
apicl(config-leaf)# exit

apicl(config)# leaf 102
apicl(config-leaf)# template route group match-rule1 tenant t1
apicl(config-route-group)# ip prefix permit 192.168.1.0/24
apicl(config-route-group)# exit
apicl(config-leaf)# template route group match-rule2 tenant t1
apicl(config-route-group)# ip prefix permit 192.168.2.0/24
apicl(config-route-group)# exit
apicl(config-leaf)# vrf context tenant t1 vrf v1
apicl(config-leaf-vrf)# route-map rp1
apicl(config-leaf-vrf-route-map)# match route group match-rule2 order 0
apicl(config-leaf-vrf-route-map-match)# exit
apicl(config-leaf-vrf-route-map)# exit
apicl(config-leaf-vrf)# route-map rp2
apicl(config-leaf-vrf-route-map)# match route group match-rule1 order 0
apicl(config-leaf-vrf-route-map-match)# exit
apicl(config-leaf-vrf-route-map)# exit
apicl(config-leaf-vrf)# exit
apicl(config-leaf)# router bgp 100
apicl(config-leaf-bgp)# vrf member tenant t1 vrf v1

```

Example: Transit Routing

```

apic1(config-leaf-bgp-vrf)# neighbor 25.25.25.2
apic1(config-leaf-bgp-vrf-neighbor)# route-map rp2 in
apic1(config-leaf-bgp-vrf-neighbor)# route-map rp1 out
apic1(config-leaf-bgp-vrf-neighbor)# exit
apic1(config-leaf-bgp-vrf)# exit
apic1(config-leaf-bgp)# exit
apic1(config-leaf)# exit

```

Step 7 Create filters (access lists) and contracts to enable the EPGs to communicate.

Example:

```

apic1(config)# tenant t1
apic1(config-tenant)# access-list http-filter
apic1(config-tenant-acl)# match ip
apic1(config-tenant-acl)# match tcp dest 80
apic1(config-tenant-acl)# exit
apic1(config-tenant)# contract httpCtrct
apic1(config-tenant-contract)# scope vrf
apic1(config-tenant-contract)# subject subj1
apic1(config-tenant-contract-subj)# access-group http-filter both
apic1(config-tenant-contract-subj)# exit
apic1(config-tenant-contract)# exit
apic1(config-tenant)# exit

```

Step 8 Configure contracts and associate them with EPGs.

Example:

```

apic1(config)# tenant t1
apic1(config-tenant)# external-l3 epɡ extnw1
apic1(config-tenant-l3ext-epɡ)# vrf member v1
apic1(config-tenant-l3ext-epɡ)# contract provider httpCtrct
apic1(config-tenant-l3ext-epɡ)# exit
apic1(config-tenant)# external-l3 epɡ extnw2
apic1(config-tenant-l3ext-epɡ)# vrf member v1
apic1(config-tenant-l3ext-epɡ)# contract consumer httpCtrct
apic1(config-tenant-l3ext-epɡ)# exit
apic1(config-tenant)# exit
apic1(config)#

```

Example: Transit Routing

This example provides a merged configuration for transit routing. The configuration is for a single tenant and VRF, with two L3Outs, on two border leaf switches, that are each connected to separate routers.

```

apic1# configure
apic1(config)# tenant t1
apic1(config-tenant)# vrf context v1
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# exit

apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# router-id 11.11.11.103
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# interface ethernet 1/3
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member tenant t1 vrf v1
apic1(config-leaf-if)# ip address 12.12.12.3/24

```

```
apic1(config-leaf-if)# exit
apic1(config-leaf)# router bgp 100
apic1(config-leaf-bgp)# vrf member tenant t1 vrf v1
apic1(config-leaf-bgp-vrf)# neighbor 15.15.15.2
apic1(config-leaf-bgp-vrf-neighbor)# exit
apic1(config-leaf-bgp-vrf)# exit
apic1(config-leaf-bgp)# exit
apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant t1 vrf v1
apic1(config-leaf-ospf-vrf)# area 0.0.0.0 loopback 40.40.40.1
apic1(config-leaf-ospf-vrf)# exit
apic1(config-leaf-ospf)# exit
apic1(config-leaf)# exit

apic1(config)# leaf 102
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# router-id 22.22.22.203
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# interface ethernet 1/3
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vrf member tenant t1 vrf v1
apic1(config-leaf-if)# ip address 23.23.23.3/24
apic1(config-leaf-if)# exit
apic1(config-leaf)# router bgp 100
apic1(config-leaf-bgp)# vrf member tenant t1 vrf v1
apic1(config-leaf-bgp-vrf)# neighbor 25.25.25.2/24
apic1(config-leaf-bgp-vrf-neighbor)# exit
apic1(config-leaf-bgp-vrf)# exit
apic1(config-leaf-bgp)# exit
apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant t1 vrf v1
apic1(config-leaf-ospf-vrf)# area 0.0.0.0 loopback 60.60.60.3
apic1(config-leaf-ospf-vrf)# exit
apic1(config-leaf-ospf)# exit
apic1(config-leaf)# exit

apic1(config)# tenant t1
apic1(config-tenant)# external-l3 epg extnw1
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# match ip 192.168.1.0/24
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# external-l3 epg extnw2
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# match ip 192.168.2.0/24
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# exit

apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# external-l3 epg extnw1
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 102
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# external-l3 epg extnw2
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# exit

apic1(config)# leaf 101
apic1(config-leaf)# template route group match-rule1 tenant t1
apic1(config-route-group)# ip prefix permit 192.168.1.0/24
apic1(config-route-group)# exit
apic1(config-leaf)# template route group match-rule2 tenant t1
```

Example: Transit Routing

```

apicl(config-route-group)# ip prefix permit 192.168.2.0/24
apicl(config-route-group)# exit
apicl(config-leaf)# vrf context tenant t1 vrf v1
apicl(config-leaf-vrf)# route-map rp1
apicl(config-leaf-vrf-route-map)# match route group match-rule1 order 0
apicl(config-leaf-vrf-route-map-match)# exit
apicl(config-leaf-vrf-route-map)# exit
apicl(config-leaf-vrf)# route-map rp2
apicl(config-leaf-vrf-route-map)# match route group match-rule2 order 0
apicl(config-leaf-vrf-route-map-match)# exit
apicl(config-leaf-vrf-route-map)# exit
apicl(config-leaf-vrf)# exit
apicl(config-leaf)# router bgp 100
apicl(config-leaf-bgp)# vrf member tenant t1 vrf v1
apicl(config-leaf-bgp-vrf)# neighbor 15.15.15.2
apicl(config-leaf-bgp-vrf-neighbor)# route-map rp1 in
apicl(config-leaf-bgp-vrf-neighbor)# route-map rp2 out
apicl(config-leaf-bgp-vrf-neighbor)# exit
apicl(config-leaf-bgp-vrf)# exit
apicl(config-leaf-bgp)# exit
apicl(config-leaf)# exit

apicl(config)# leaf 102
apicl(config-leaf)# template route group match-rule1 tenant t1
apicl(config-route-group)# ip prefix permit 192.168.1.0/24
apicl(config-route-group)# exit
apicl(config-leaf)# template route group match-rule2 tenant t1
apicl(config-route-group)# ip prefix permit 192.168.2.0/24
apicl(config-route-group)# exit
apicl(config-leaf)# vrf context tenant t1 vrf v1
apicl(config-leaf-vrf)# route-map rp1
apicl(config-leaf-vrf-route-map)# match route group match-rule1 order 0
apicl(config-leaf-vrf-route-map-match)# exit
apicl(config-leaf-vrf-route-map)# exit
apicl(config-leaf-vrf)# route-map rp2
apicl(config-leaf-vrf-route-map)# match route group match-rule2 order 0
apicl(config-leaf-vrf-route-map-match)# exit
apicl(config-leaf-vrf-route-map)# exit
apicl(config-leaf-vrf)# exit
apicl(config-leaf)# router bgp 100
apicl(config-leaf-bgp)# vrf member tenant t1 vrf v1
apicl(config-leaf-bgp-vrf)# neighbor 25.25.25.2
apicl(config-leaf-bgp-vrf-neighbor)# route-map rp2 in
apicl(config-leaf-bgp-vrf-neighbor)# route-map rp1 out
apicl(config-leaf-bgp-vrf-neighbor)# exit
apicl(config-leaf-bgp-vrf)# exit
apicl(config-leaf-bgp)# exit
apicl(config-leaf)# exit

apicl(config)# tenant t1
apicl(config-tenant)# access-list http-filter
apicl(config-tenant-acl)# match ip
apicl(config-tenant-acl)# match tcp dest 80
apicl(config-tenant-acl)# exit
apicl(config-tenant)# contract httpCtrct
apicl(config-tenant-contract)# scope vrf
apicl(config-tenant-contract)# subject http-subj
apicl(config-tenant-contract-subj)# access-group http-filter both
apicl(config-tenant-contract-subj)# exit
apicl(config-tenant-contract)# exit
apicl(config-tenant)# exit

apicl(config)# tenant t1
apicl(config-tenant)# external-l3 epg extnw1

```



```
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# contract provider httpCtrct
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# external-l3 epg extnw2
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# contract consumer httpCtrct
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# exit
apic1(config)#
```




CHAPTER 8

Configuring Cisco ACI QoS

This chapter contains the following sections:

- [L3Outs QoS, on page 351](#)
- [CoS Preservation, on page 353](#)
- [Multipod QoS, on page 354](#)
- [Translating QoS Ingress Markings to Egress Markings, on page 356](#)

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 Contract for L3Out Using NX-OS Style CLI

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 CLI, on page 352](#) instead.

Step 1 Configure the VRF for egress mode and enable policy enforcement to support QoS priority enforcement on the L3Out.

Example:

```

apic1# configure
apic1(config)# tenant t1
apic1(config-tenant)# vrf context v1
apic1(config-tenant-vrf)# contract enforce egress
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# exit
apic1(config)#

```

Step 2 Configure QoS.

When creating filters (`access-list`), include the **match dscp** command with target DSCP level.

When configuring contracts, include the QoS class for traffic ingressing on the L3Out. Alternatively, you can define a target DSCP value. QoS policies are supported on either the contract or the subject

VRF enforcement must 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.

Example:

```

apic1(config)# tenant t1
apic1(config-tenant)# access-list http-filter
apic1(config-tenant-acl)# match ip
apic1(config-tenant-acl)# match tcp dest 80
apic1(config-tenant-acl)# match dscp EF
apic1(config-tenant-acl)# exit
apic1(config-tenant)# contract httpContract
apic1(config-tenant-contract)# scope vrf
apic1(config-tenant-contract)# qos-class level1
apic1(config-tenant-contract)# subject http-subject
apic1(config-tenant-contract-subj)# access-group http-filter both
apic1(config-tenant-contract-subj)# exit
apic1(config-tenant-contract)# exit
apic1(config-tenant)# exit
apic1(config)#

```

Configuring QoS Directly on L3Out Using CLI

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:

```

interface vlan 19
  vrf member tenant DT vrf dt-vrf

```

```
ip address 107.2.1.252/24
description 'SVI19'
service-policy type qos VrfQos006 // for custom QoS attachment
set qos-class level6 // for set QoS priority
exit
```

Step 2 Configure QoS priorities for a sub-interface.

Example:

```
interface ethernet 1/48.10
  vrf member tenant DT vrf inter-tentant-ctx2 l3out L4_E48_inter_tenant
  ip address 210.2.0.254/16
  service-policy type qos vrfQos002
  set qos-class level5
```

Step 3 Configure QoS priorities for a routed outside.

Example:

```
interface ethernet 1/37
  no switchport
  vrf member tenant DT vrf dt-vrf l3out L2E37
  ip address 30.1.1.1/24
  service-policy type qos vrfQos002
  set qos-class level5
  exit
```

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 of the VXLAN 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 355](#).

Enable Class Of Service (CoS) Preservation Using NX-OS Style CLI

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.

Step 1 Enter configuration mode.

Example:

```
apic1# configure
```

Step 2 Enables CoS preservation.

Example:

```
apic1(config)# qos preserve cos
```

Multipod QoS

Creating DSCP Translation Policy Using NX-OS Style CLI

This section describes how to create a DSCP translation policy to guarantee QoS Level settings across multiple PODs connected by an IPN.

Step 1 Enters configuration mode.

Example:

```
apic1# configure
```

Step 2 Enters tenant configuration mode for the `infra` tenant.

Example:

```
apic1(config)# tenant infra
```

Step 3 Create the DSCP translation map.

Example:

```
apic1(config-tenant)# qos dscp-map default
```

Step 4 Configure the DSCP translation mappings.

Note All mappings must be unique within a DSCP translation map and you must not map any QoS level to `cs6`.

Example:

```
apic1(config-qos-cmap# set dscp-code control CS3
apic1(config-qos-cmap# set dscp-code span CS5
apic1(config-qos-cmap# set dscp-code level1 CS0
apic1(config-qos-cmap# set dscp-code level2 CS1
apic1(config-qos-cmap# set dscp-code level3 CS2
apic1(config-qos-cmap# set dscp-code level4 CS3
```

```

apic1(config-qos-cmap# set dscp-code level15 CS4
apic1(config-qos-cmap# set dscp-code level16 CS5
apic1(config-qos-cmap# set dscp-code policy CS4
apic1(config-qos-cmap# set dscp-code traceroute CS5

```

Step 5 Enable the DSCP translation.

Example:

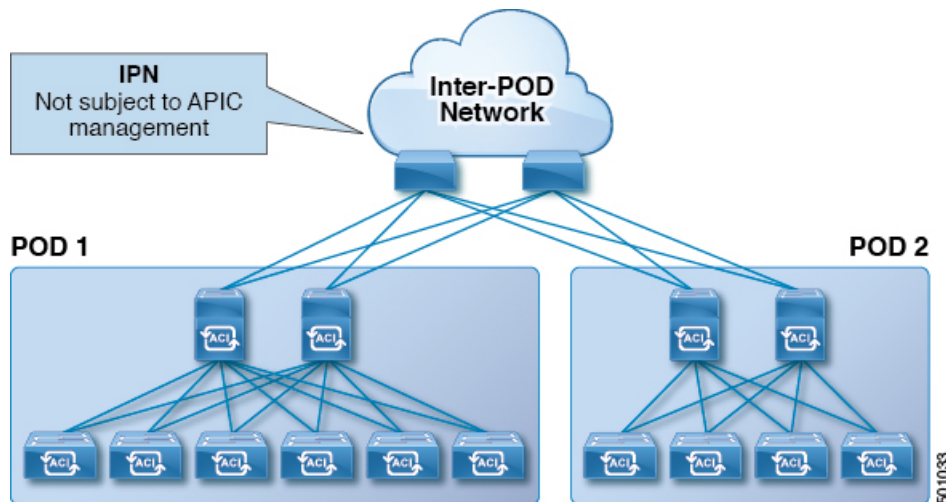
```
apic1(config-qos-cmap) # no shutdown
```

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 353](#) instead.

Figure 27: 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.

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 353](#).

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 NX-OS Style CLI

This section describes how to create a custom QoS policy and associate it with an EPG using the NX-OS style CLI.

Before you begin

You must have created the tenant, application, and EPGs that will consume the custom QoS policy.

Step 1 Enter configuration mode.

Example:

```
apic1# configure
```

Step 2 Enter tenant configuration mode.

Example:

```
apic1(config)# tenant <tenant-name>
```


Step 3 Create QoS policy.

Example:

```
apic1(config-tenant)# policy-map type qos <qos-policy-name>
```

Step 4 Set DCSP range and target QoS priority level.

Example:

Example:

```
apic1(config-tenant-pmap-qos)# match dscp AF23 AF31 set-cos 6
```

Step 5 Return to tenant configuration mode.

Example:

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

Step 6 Create or edit an application profile.

Example:

```
apic1(config-tenant)# application <application-name>
```

Step 7 Create or edit an EPG in the application profile.

To create a normal EPG:

Example:

```
apic1(config-tenant-app)# epg <epg-name>
```

To create an external Layer-2 EPG:

Example:

```
apic1(config-tenant)# external-l2 epg <ext-l2-epg-name>
```

Step 8 Associate the QoS policy with the EPG.

The system prompt may be different depending on whether you create a normal EPG or an external EPG.

Example:

```
apic1(config-tenant-app-epg)# service-policy <qos-policy-name>
```

Step 9 Return to the tenant configuration mode.

Example:

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




CHAPTER 9

Configuring Management Interfaces

- [Configuring Out-of-Band Management Access, on page 359](#)
- [Configuring Inband Management Access, on page 362](#)

Configuring Out-of-Band Management Access

To configure out-of-band (OOB) management access for controllers, leaf switches, or spine switches, these steps must be performed:

- Configure the OOB management IP address and gateway on the management interface
- Allow access from the necessary external subnets
- Allow the necessary protocols on the management ports

Before you begin

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

SUMMARY STEPS

1. **configure**
2. **{controller *apic-number-or-range* | switch *node-id*[-*node-id-or-range*]}**
3. **interface mgmt0**
4. **ip address *addr/mask* gateway *addr***
5. **exit**
6. **exit**
7. **tenant mgmt**
8. **external-l3 epg default oob-mgmt**
9. **match ip *addr/mask***
10. **exit**
11. **access-list oob-default**
12. **match tcp dest 443**
13. **match tcp dest 22**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	{controller <i>apic-number-or-range</i> switch <i>node-id</i>[-<i>node-id-or-range</i>]} Example: <code>apic1(config)# controller 1-3</code>	Specifies the controller or switch to be configured. You can enter a range of controllers or switches using dashes or commas.
Step 3	interface mgmt0 Example: <code>apic1(config-controller)# interface mgmt0</code>	The mgmt0 interface provides out-of-band management, which enables you to manage the device by its IPv4 address.
Step 4	ip address <i>addr/mask</i> gateway <i>addr</i> Example: <code>apic1(config-controller-if)# ip address-range 172.23.48.16/21 gateway 172.23.48.1</code>	Configures the IP address and gateway for OOB management. If you specified more than one controller or switch, the command becomes ip address-range and IP addresses are assigned sequentially beginning with the address specified in this command. Note The APIC management interface does not support an IPv6 address and cannot connect to an external IPv6 server through this interface.
Step 5	exit Example: <code>apic1(config-controller-if)# exit</code>	
Step 6	exit Example: <code>apic1(config-controller)# exit</code>	
Step 7	tenant mgmt Example: <code>apic1(config)# tenant mgmt</code>	System Management policies are configured under a special tenant called mgmt .
Step 8	external-l3 epg default oob-mgmt Example: <code>apic1(config-tenant)# external-l3 epg default oob-mgmt</code>	Enters the configuration mode of the out-of-band management EPG.
Step 9	match ip <i>addr/mask</i> Example: <code>apic1(config-tenant-l3ext-epg)# match ip 192.0.20.0/24</code>	Provides access control for out-of-band management interface to external management subnets.

	Command or Action	Purpose
Step 10	exit Example: <code>apic1(config-tenant-l3ext-epg)# exit</code>	
Step 11	access-list oob-default Example: <code>apic1(config-tenant)# access-list oob-default</code>	Configures the access list filter for the OOB default policy.
Step 12	match tcp dest 443 Example: <code>apic1(config-tenant-acl)# match tcp dest 443</code>	Allows access on the management interface for HTTPS traffic (TCP/443).
Step 13	match tcp dest 22 Example: <code>apic1(config-tenant-acl)# match tcp dest 22</code>	Allows access on the management interface for SSH traffic (TCP/22).

Examples

This example shows how to configure out-of-band management access for three APIC controllers. In this example, the three controllers are assigned sequential IP addresses, with controller 1 at 172.23.48.16/21, controller 2 at 172.23.48.17/21, and controller 3 at 172.23.48.18/21.

```
apic1# configure
apic1(config)# controller 1-3
apic1(config-controller)# interface mgmt0
apic1(config-controller-if)# ip address-range 172.23.48.16/21 gateway 172.23.48.1
apic1(config-controller-if)# exit
apic1(config-controller)# exit
apic1(config)# tenant mgmt
apic1(config-tenant)# external-l3 epg default oob-mgmt
apic1(config-tenant-l3ext-epg)# match ip 192.0.20.0/24
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# access-list oob-default
apic1(config-tenant-acl)# match tcp dest 443
apic1(config-tenant-acl)# match tcp dest 22
```

This example shows how to configure out-of-band management access for a leaf or spine switch.

```
apic1# configure
apic1(config)# switch 101
apic1(config-switch)# interface mgmt0
apic1(config-switch-if)# ip address 172.23.48.101/21 gateway 172.23.48.1
```

Configuring Inband Management Access

Configuring Inband Management Access to a Switch from an Outside Network

To configure inband (IB) management access for leaf switches or spine switches, these steps must be performed:

- Configure the inband management IP address and gateway on the inband management interface
- Create or specify a VLAN domain for external inband connectivity
- Add the external management station interface to the VLAN domain
- Allow the necessary protocols on the management ports

SUMMARY STEPS

1. **configure**
2. **switch** *switch-id-or-range*
3. **interface inband-mgmt0**
4. **ip address** *addr/mask gateway addr*
5. **exit**
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apicl# configure</code>	Enters configuration mode.
Step 2	switch <i>switch-id-or-range</i> Example: <code>apicl(config)# switch 101</code>	Specifies the switch to be configured. You can enter a range of switches using dashes or commas.
Step 3	interface inband-mgmt0 Example: <code>apicl(config-switch)# interface inband-mgmt0</code>	The inband-mgmt0 interface provides inband management.
Step 4	ip address <i>addr/mask gateway addr</i> Example: <code>apicl(config-switch-if)# ip address 10.13.1.1/24 gateway 10.13.1.254</code>	Configures the IP address and gateway for inband management. If you specified more than one switch, the command becomes ip address-range and IP addresses are assigned sequentially beginning with the address specified in this command.
Step 5	exit Example: <code>apicl(config-switch-if)# exit</code>	

	Command or Action	Purpose
Step 6	exit Example: <code>apic1(config-switch)# exit</code>	

Examples

This example shows how to configure inband management for a switch from a management station on an external network..

```
apic1# configure
apic1(config)# switch 101
apic1(config-switch)# interface inband-mgmt0
apic1(config-switch-if)# ip address 10.13.1.1/24 gateway 10.13.1.254
apic1(config-switch-if)# exit
apic1(config-switch)# exit
```

What to do next

- Configure inband (IB) management connectivity to the management station.
- Allow the necessary protocols (HTTPS and SSH) on the inbound management port.

Configuring Inband Management Access to a Controller from an Outside Network

To configure inband (IB) management access for controllers, these steps must be performed:

- Configure the inband management IP address and gateway on the inband management interface
- Create a VLAN domain for external inband connectivity
- Allow the VLAN on the port connected to the controller

SUMMARY STEPS

1. **configure**
2. **controller** *controller-id-or-range*
3. **interface inband-mgmt0**
4. **ip address** *addr/mask gateway addr*
5. **vlan** *vlan-id*
6. **exit**
7. **exit**
8. **vlan-domain** *domain-name*
9. **vlan** *vlan-id*
10. **exit**

11. `leaf node-id`
12. `interface slot/port`
13. `vlan-domain member apic-inband`
14. `exit`
15. `exit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code> Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	<code>controller controller-id-or-range</code> Example: <code>apic1(config)# controller 1-3</code>	Specifies the controller to be configured. You can enter a range of controllers using dashes or commas.
Step 3	<code>interface inband-mgmt0</code> Example: <code>apic1(config-controller)# interface inband-mgmt0</code>	The inband-mgmt0 interface provides inband management.
Step 4	<code>ip address addr/mask gateway addr</code> Example: <code>apic1(config-controller-if)# ip address-range 10.13.1.1/24 gateway 10.13.1.254</code>	Configures the IP address and gateway for inband management. If you specified more than one controller or switch, the command becomes ip address-range and IP addresses are assigned sequentially beginning with the address specified in this command.
Step 5	<code>vlan vlan-id</code> Example: <code>apic1(config-controller-if)# vlan 10</code>	Assigns a controller VLAN which is enabled on the port connected to the controller. For multiple controllers, all controllers must use the same VLAN.
Step 6	<code>exit</code> Example: <code>apic1(config-controller-if)# exit</code>	
Step 7	<code>exit</code> Example: <code>apic1(config-controller)# exit</code>	
Step 8	<code>vlan-domain domain-name</code> Example: <code>apic1(config)# vlan-domain apic-inband</code>	Creates and enters the configuration mode for the VLAN domain.
Step 9	<code>vlan vlan-id</code> Example: <code>apic1(config-vlan)# vlan 10</code>	Assigns the controller VLAN to the VLAN domain.

	Command or Action	Purpose
Step 10	exit Example: <code>apic1(config-vlan)# exit</code>	Returns to global configuration mode.
Step 11	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 102</code>	Specifies the leaf switch to which the controller connected.
Step 12	interface <i>slot/port</i> Example: <code>apic1(config-leaf)# interface eth 1/1</code>	Specifies the port to which the controller is connected.
Step 13	vlan-domain member apic-inband Example: <code>apic1(config-leaf-if)# vlan-domain member apic-inband</code>	Configures controller connectivity to inband management.
Step 14	exit Example: <code>apic1(config-leaf-if)# exit</code>	
Step 15	exit Example: <code>apic1(config-leaf)# exit</code>	

Examples

This example shows how to configure inband management for a controller from a management station on an external network. APIC controller 1 is connected to port Ethernet 1/1 on Leaf 101, and VLAN 10 is used for the controller's inband connectivity.

```
apic1# configure
apic1(config)# controller 1-3
apic1(config-controller)# interface inband-mgmt0
apic1(config-controller-if)# ip address-range 10.13.1.1/24 gateway 10.13.1.254
apic1(config-controller-if)# vlan 10
apic1(config-controller-if)# exit
apic1(config-controller)# exit

# CREATE A VLAN DOMAIN FOR THE APIC INBAND VLAN
apic1(config)# vlan-domain apic-inband
apic1(config-vlan)# vlan 10
apic1(config-vlan)# exit

# ALLOW THE VLAN ON THE PORT CONNECTED TO THE CONTROLLER
apic1(config)# leaf 101
apic1(config-leaf)# interface eth 1/1
apic1(config-leaf-if)# vlan-domain member apic-inband
```

```
apic1(config-leaf-if) # exit
apic1(config-leaf) # exit
```

What to do next

- Configure inband (IB) management connectivity to the management station.
- Allow the necessary protocols (HTTPS and SSH) on the inbound management port.

Configuring Inband Management Connectivity to the Management Station

To configure inband (IB) management connectivity to the management station, these steps must be performed:

- Create or specify a VLAN domain for external inband connectivity
- Add the external management station interface to the VLAN domain

SUMMARY STEPS

1. **configure**
2. **vlan-domain** *domain-name*
3. **vlan** *vlan-id*
4. **exit**
5. **leaf** *node-id*
6. **interface** *slot/port*
7. **vlan-domain member** **external-inband**
8. **switchport trunk allowed vlan** *vlan-id* **inband-mgmt** *gateway-ip/mask*
9. **exit**
10. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	vlan-domain <i>domain-name</i> Example: <code>apic1(config)# vlan-domain external-inband</code>	Creates and enters the configuration mode for the VLAN domain.
Step 3	vlan <i>vlan-id</i> Example: <code>apic1(config-vlan)# vlan 11</code>	Assigns a VLAN to the domain.
Step 4	exit Example:	Returns to global configuration mode.

	Command or Action	Purpose
	<code>apic1(config-vlan)# exit</code>	
Step 5	leaf <i>node-id</i> Example: <code>apic1(config)# leaf 102</code>	Specifies the leaf switch to which the management station is connected.
Step 6	interface <i>slot/port</i> Example: <code>apic1(config-leaf)# interface eth 1/2</code>	Specifies the port to which the management station is connected.
Step 7	vlan-domain member external-inband Example: <code>apic1(config-leaf-if)# vlan-domain member external-inband</code>	Configures external layer2 connectivity to inband management.
Step 8	switchport trunk allowed vlan <i>vlan-id</i> inband-mgmt gateway-ip/mask Example: <code>apic1(config-leaf-if)# switchport trunk allowed vlan 11 inband-mgmt 179.10.1.254/24</code>	Configures external layer2 connectivity to inband management. The specified IP address is the gateway address used by the external management station and the gateway functionality is provided by the ACI fabric.
Step 9	exit Example: <code>apic1(config-leaf-if)# exit</code>	
Step 10	exit Example: <code>apic1(config-leaf)# exit</code>	

Examples

This example shows how to configure inband management connectivity to the management station.

```
# CREATE A VLAN DOMAIN FOR EXTERNAL CONNECTIVITY TO INBAND MANAGEMENT
apic1# configure
apic1(config)# vlan-domain external-inband
apic1(config-vlan)# vlan 11
apic1(config-vlan)# exit

# CONFIGURE LAYER 2 CONNECTIVITY FROM THE MANAGEMENT STATION INTERFACE TO INBAND MANAGEMENT
apic1(config)# leaf 102
apic1(config-leaf)# interface eth 1/2
apic1(config-leaf-if)# vlan-domain member external-inband
apic1(config-leaf-if)# switchport trunk allowed vlan 11 inband-mgmt 179.10.1.254/24
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

What to do next

- Allow the necessary protocols (HTTPS and SSH) on the inbound management port.

Configuring Inband Management Contract to Open HTTPS/SSH Ports

SUMMARY STEPS

1. **configure**
2. **tenant mgmt**
3. **access-list inband-default**
4. **match tcp dest 443**
5. **match tcp dest 22**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters configuration mode.
Step 2	tenant mgmt Example: apicl(config)# tenant mgmt	System Management policies are configured under a special tenant called mgmt .
Step 3	access-list inband-default Example: apicl(config-tenant)# access-list inband-default	Configures the access list filter for the inband default policy.
Step 4	match tcp dest 443 Example: apicl(config-tenant-acl)# match tcp dest 443	Allows access on the management interface for HTTPS traffic (TCP/443).
Step 5	match tcp dest 22 Example: apicl(config-tenant-acl)# match tcp dest 22	Allows access on the management interface for SSH traffic (TCP/22).

Examples

This example shows how to allow HTTPS and SSH access to the inband management port.

```

apicl# configure
apicl(config)# tenant mgmt
apicl(config-tenant)# access-list inband-default
apicl(config-tenant-acl)# match tcp dest 443
apicl(config-tenant-acl)# match tcp dest 22

```

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




CHAPTER 10

Configuring Security

- [About Security Configuration, on page 371](#)
- [Configuring AAA, on page 372](#)
- [Configuring Security Servers, on page 376](#)
- [Configuring the Password Policy, on page 383](#)
- [Configuring Users, on page 385](#)
- [Configuring Public Key Infrastructure, on page 390](#)
- [Configuring Communication Policies, on page 396](#)
- [Configuring AES Encryption, on page 401](#)
- [Configuring Fabric Secure Mode, on page 402](#)
- [Configuring COOP Authentication, on page 404](#)
- [Configuring FIPS, on page 405](#)
- [Configuring Control Plane Policing, on page 406](#)
- [Configuring First Hop Security, on page 410](#)
- [Configuring 802.1x, on page 417](#)

About Security Configuration

Access control is the way you control who is allowed access to the network server and what services they are allowed to use once they have access. Authentication, authorization, and accounting (AAA) network security services provide the primary framework through which you set up access control on APIC.

Overview of the AAA Configuration

To configure security on APIC using AAA, follow this process:

1. To use a separate security server, configure security protocol parameters using the **radius-server**, **ldap-server**, or **tacacs-server** configuration commands.
2. Define the method lists for authentication by using an **aaa authentication** command.
3. Apply the method lists to a particular interface or line, if required.
4. (Optional) Configure authorization using the **aaa authentication** command.

Login Authentication Using a Local Password

Use the **aaa authentication login** command with the *method* argument to specify that APIC will use the local username database for authentication. For example, to specify the local username database as the method of user authentication at login when no other method list has been defined, enter the following commands:

```
apicl# configure
apicl(config)# aaa authentication login default
apicl(config-default)# realm local
```

For information about adding users into the local username database, refer to the section “Configuring a Locally Authenticated User.”

Login Authentication Using a Remote Server

Use the **aaa authentication login** command with the server radius/tacacs/ldap method to specify RADIUS/TACACS+/LDAP as the login authentication method. For example, to specify RADIUS as the method of user authentication at login when no other method list has been defined, enter the following commands:

```
apicl# configure
apicl(config)# aaa authentication login default
apicl(config-default)# realm radius
```

Before you can use RADIUS as the login authentication method, you need to enable communication with the RADIUS security server, same is true for TACACS+ or LDAP. For more information about establishing communication with a remote security server, see the appropriate chapter:

- "Configuring a RADIUS Server"
- "Configuring a TACACS+ Server"
- "Configuring an LDAP Server"

Configuring AAA

SUMMARY STEPS

1. **configure**
2. **aaa authentication login console**
3. **[no] realm {ldap | local | radius | tacacs}**
4. **[no] group *group-name***
5. **exit**
6. **aaa authentication login default**
7. **[no] realm {ldap | local | radius | tacacs}**
8. **[no] group *group-name***
9. **exit**
10. **aaa authentication login domain {*domain-name* | fallback}**
11. **[no] realm {ldap | local | none | radius | tacacs}**

12. **[no] group** *group-name*
13. **exit**
14. **aaa banner** *text*
15. **aaa group** {**ldap** | **radius** | **tacacs**} *group-name*
16. **[no] server** {*ip-address* | *hostname*} **priority** *priority-number*
17. **exit**
18. **aaa scvmm-certificate** *certificate-name*
19. **aaa user default-role** {**assign-default-role** | **no-login**}
20. **show aaa authentication**
21. **show aaa groups**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	aaa authentication login console Example: <code>apic1(config)# aaa authentication login console</code>	Enters console configuration mode for users accessing APIC through the console.
Step 3	[no] realm { ldap local radius tacacs } Example: <code>apic1(config-console)# realm radius</code>	Specifies the authentication method.
Step 4	[no] group <i>group-name</i> Example: <code>apic1(config-console)# group radiusGroup5</code>	Specifies an authentication server group.
Step 5	exit Example: <code>apic1(config-console)# exit</code>	Returns to global configuration mode.
Step 6	aaa authentication login default Example: <code>apic1(config)# aaa authentication login default</code>	Enters the configuration mode for default login authentication.
Step 7	[no] realm { ldap local radius tacacs } Example: <code>apic1(config-default)# realm radius</code>	Specifies the authentication method.
Step 8	[no] group <i>group-name</i> Example: <code>apic1(config-default)# group radiusGroup</code>	Specifies an authentication server group.

	Command or Action	Purpose
Step 9	exit Example: <code>apic1(config-default)# exit</code>	Returns to global configuration mode.
Step 10	aaa authentication login domain { <i>domain-name</i> fallback } Example: <code>apic1(config)# aaa authentication login domain cisco</code>	Enters the configuration mode for default login authentication. A login domain specifies the authentication domain for a user.
Step 11	[no] realm { ldap local none radius tacacs } Example: <code>apic1(config-domain)# realm radius</code>	Specifies the authentication method.
Step 12	[no] group <i>group-name</i> Example: <code>apic1(config-domain)# group radiusGroup</code>	Specifies an authentication server group.
Step 13	exit Example: <code>apic1(config-domain)# exit</code>	Returns to global configuration mode.
Step 14	aaa banner <i>text</i> Example: <code>apic1(config)# aaa banner 'Welcome to APIC'</code>	Specifies the informational banner to be displayed before the user login. The banner must be contained in single quotes.
Step 15	aaa group { ldap radius tacacs } <i>group-name</i> Example: <code>apic1(config)# aaa group radius radiusGroup</code>	Creates or configures an authentication server group.
Step 16	[no] server { <i>ip-address</i> <i>hostname</i> } priority <i>priority-number</i> Example: <code>apic1(config-radius)# server 192.0.20.71 priority 2</code>	Adds a server to the authentication server group and specifies its priority within the server group. The priority can be between 0 and 17.
Step 17	exit Example: <code>apic1(config-radius)# exit</code>	Returns to global configuration mode.
Step 18	aaa scvmm-certificate <i>certificate-name</i> Example: <code>apic1(config)# aaa scvmm-certificate myScvmmCert</code>	Specifies an SCVMM certificate. See the <i>Cisco ACI Virtualization Guide</i> .

	Command or Action	Purpose
Step 19	aaa user default-role {assign-default-role no-login} Example: <pre>apic1(config)# aaa user default-role assign-default-role</pre>	Specifies how to respond when remote users who do not have a user role attempt to log in to APIC. The action can be either of these options: • assign-default-role—Remote users who do not have a user role are assigned a default role. • no-login—Remote users who do not have a user role cannot log in.
Step 20	show aaa authentication Example: <pre>apic1(config)# show aaa authentication</pre>	Displays configured AAA methods.
Step 21	show aaa groups Example: <pre>apic1(config)# show aaa groups</pre>	Displays configured AAA server groups.

Examples

This example shows how to configure AAA.

```
apic1# configure terminal
apic1(config)# aaa authentication login console
apic1(config-console)# realm local
apic1(config-console)# exit
apic1(config)# aaa authentication login default
apic1(config-default)# realm radius
apic1(config-default)# group radiusGroup5
apic1(config-default)# exit
apic1(config)# aaa authentication login domain cisco
apic1(config-domain)# realm none
apic1(config-domain)# exit
apic1(config)# aaa banner 'Welcome to APIC'
apic1(config)# aaa group radius radiusGroup
apic1(config-radius)# server 192.0.20.71 priority 2
apic1(config-radius)# exit
apic1(config)# aaa user default-role assign-default-role
apic1(config)# show aaa authentication
Default : radius
Console : local

apic1(config)# show aaa groups
Total number of Groups : 1

RadiusGroups : radiusGroup5
TacacsGroups :
LdapGroups   :
```

Configuring Security Servers

Configuring a RADIUS Server

SUMMARY STEPS

1. **configure**
2. **[no] radius-server retries** *count*
3. **[no] radius-server timeout** *seconds*
4. **[no] radius-server host** *{ip-address | hostname}*
5. (Optional) **[no] retries** *count*
6. (Optional) **[no] timeout** *seconds*
7. (Optional) **[no] descr** *text*
8. **[no] key** *key-value*
9. **[no] port** *port-number*
10. **[no] protocol** *{chap | mschap | pap}*
11. **exit**
12. **show radius-server**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] radius-server retries <i>count</i> Example: <code>apic1(config)# radius-server retries 1</code>	Specifies how many times APIC transmits each RADIUS request to the server before giving up. The range is 0 to 5. In the global configuration mode, this command applies to all RADIUS servers unless overridden in the specific RADIUS host configuration.
Step 3	[no] radius-server timeout <i>seconds</i> Example: <code>apic1(config)# radius-server timeout 5</code>	Specifies the number of seconds APIC waits for a reply to a RADIUS request before retransmitting the request. In the global configuration mode, this command applies to all RADIUS servers unless overridden in the specific RADIUS host configuration.
Step 4	[no] radius-server host <i>{ip-address hostname}</i> Example: <code>apic1(config)# radius-server host 192.0.20.71</code>	Specifies the IP address or hostname of the RADIUS server.

	Command or Action	Purpose
Step 5	(Optional) [no] retries <i>count</i> Example: <code>apic1(config-host)# retries 2</code>	For this RADIUS server, specifies how many times APIC transmits each RADIUS request to the server before giving up. The range is 0 to 5. If no retry count is set, the global value is used.
Step 6	(Optional) [no] timeout <i>seconds</i> Example: <code>apic1(config-host)# timeout 3</code>	For this RADIUS server, specifies the number of seconds APIC waits for a reply to a RADIUS request before retransmitting the request. If no timeout is set, the global value is used.
Step 7	(Optional) [no] descr <i>text</i> Example: <code>apic1(config-host)# descr "My primary RADIUS server"</code>	Provides descriptive information about this RADIUS server. The text can be up to 128 alphanumeric characters. If the text contains spaces, it must be enclosed by single or double quotes.
Step 8	[no] key <i>key-value</i> Example: <code>apic1(config-host)# key myRaDiUSpassWoRd</code>	Specifies the shared secret text string used between APIC and this RADIUS server for authentication. The key can be up to 32 characters.
Step 9	[no] port <i>port-number</i> Example: <code>apic1(config-host)# port 1812</code>	Specifies a UDP port on this RADIUS server to be used solely for authentication.
Step 10	[no] protocol {chap mschap pap} Example: <code>apic1(config-host)# protocol pap</code>	Specifies the RADIUS server protocol for authentication.
Step 11	exit Example: <code>apic1(config-host)#</code>	Returns to global configuration mode.
Step 12	show radius-server Example: <code>apic1(config)# show radius-server</code>	(Optional) Displays the RADIUS server information.

Examples

This example shows how to configure RADIUS settings globally and on one RADIUS server.

```
apic1# configure
apic1(config)# radius-server retries 1
apic1(config)# radius-server timeout 5
apic1(config)# radius-server host 192.0.20.71
apic1(config-host)# retries 2
apic1(config-host)# timeout 3
apic1(config-host)# descr "My primary RADIUS server"
```

```

apic1(config-host)# key myRaDiUSpassWoRd
apic1(config-host)# port 1812
apic1(config-host)# protocol pap
apic1(config-host)# exit
apic1(config)# show radius-server
timeout : 5
retries : 1

Total number of servers : 1

Hostname : 192.0.20.71
Port      : 1812
Protocol  : pap
Timeout   : 3
Retries   : 2
User      : test
Descr     : My primary RADIUS server
    
```

Configuring a TACACS+ Server

SUMMARY STEPS

1. **configure**
2. **[no] tacacs-server retries** *count*
3. **[no] tacacs-server timeout** *seconds*
4. **[no] tacacs-server host** *{ip-address | hostname}*
5. (Optional) **[no] retries** *count*
6. **[no] key**
7. **[no] port** *port-number*
8. **[no] protocol** *{chap | mschap | pap}*
9. **exit**
10. **show tacacs-server**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	[no] tacacs-server retries <i>count</i> Example: apic1(config)# tacacs-server retries 1	Specifies how many times APIC transmits each TACACS+ request to the server before giving up. The range is 0 to 5. In the global configuration mode, this command applies to all TACACS+ servers unless overridden in the specific TACACS+ host configuration.
Step 3	[no] tacacs-server timeout <i>seconds</i> Example:	Specifies the number of seconds APIC waits for a reply to a TACACS+ request before retransmitting the request.

	Command or Action	Purpose
	<code>apic1(config)# tacacs-server timeout 5</code>	In the global configuration mode, this command applies to all TACACS+ servers unless overridden in the specific TACACS+ host configuration.
Step 4	[no] tacacs-server host {ip-address hostname} Example: <code>apic1(config)# tacacs-server host 192.0.20.71</code>	Specifies the IP address or hostname of the TACACS+ server.
Step 5	(Optional) [no] retries count Example: <code>apic1(config-host)# retries 2</code>	For this TACACS+ server, specifies how many times APIC transmits each TACACS+ request to the server before giving up. The range is 0 to 5. If no retry count is set, the global value is used.
Step 6	[no] key Example: <code>apic1(config-host)# key</code> Enter key: myTacAcSpaSSWoRd Enter key again: myTacAcSpaSSWoRd	Specifies the shared secret text string used between APIC and this TACACS+ server for authentication. The key can be up to 32 characters. For increased security, entering the key value is interactive.
Step 7	[no] port port-number Example: <code>apic1(config-host)# port 49</code>	Specifies a UDP port on this TACACS+ server to be used for TACACS+ accounting messages.
Step 8	[no] protocol {chap mschap pap} Example: <code>apic1(config-host)# protocol pap</code>	Specifies the TACACS+ server protocol for authentication.
Step 9	exit Example: <code>apic1(config-host)#</code>	Returns to global configuration mode.
Step 10	show tacacs-server Example: <code>apic1(config)# show tacacs-server</code>	(Optional) Displays the TACACS+ server information.

Examples

This example shows how to configure TACACS+ settings globally and on one TACACS+ server.

```
apic1# configure
apic1(config)# tacacs-server retries 1
apic1(config)# tacacs-server timeout 5
apic1(config)# tacacs-server host 192.0.20.72
apic1(config-host)# retries 2
apic1(config-host)# timeout 3
apic1(config-host)# key myTaCaCspassWoRd
apic1(config-host)# port 49
```

```

apicl(config-host)# protocol pap
apicl(config-host)# exit
apicl(config)# show tacacs-server
timeout : 5
retries : 1

Total number of servers : 1

Hostname : 192.0.20.72
Port      : 1812
Protocol  : pap
Timeout   : 3
Retries   : 2
User      : test
    
```

Configuring an LDAP Server

Some **ldap-server** commands can be entered in either the global configuration mode or in the configuration mode for a specific LDAP host. In the global configuration mode, the command applies to all LDAP servers unless overridden in the specific LDAP host configuration.

SUMMARY STEPS

1. **configure terminal**
2. **[no] ldap-server host** *{ip-address | hostname}*
3. **[no] ldap-server attribute** *attribute-name*
4. **[no] ldap-server basedn**
5. **[no] ldap-server binddn**
6. **[no] ldap-server retries** *count*
7. **[no] ldap-server timeout** *seconds*
8. **[no] ldap-server filter** *filter-expression*
9. **[no] key** *key-value*
10. **[no] port** *port-number*
11. (Optional) **[no] retries** *count*
12. **[no] enable-ssl**
13. **[no] ssl-validation-level** *[permissive | strict]*
14. (Optional) **[no] timeout** *seconds*
15. **exit**
16. **show ldap-server**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure terminal Example: switch# configure terminal	Enters global configuration mode.

	Command or Action	Purpose
Step 2	[no] ldap-server host <i>{ip-address hostname}</i> Example: <code>apic1(config)# ldap-server host 192.0.20.73</code>	Specifies the IP address or hostname of the LDAP server and enters the configuration mode of that server.
Step 3	[no] ldap-server attribute <i>attribute-name</i> Example: <code>apic1(config-host)# ldap-server attribute memberOf</code>	Specifies an LDAP endpoint attribute to be used as the CiscoAVPair. In the global configuration mode, this command applies to all LDAP servers unless overridden in the specific LDAP host configuration.
Step 4	[no] ldap-server basedn Example: <code>apic1(config-host)# ldap-server basedn DC=sampledesign,DC=com</code>	Specifies the location in the LDAP hierarchy where the server should begin searching when it receives an authorization request. This can be a string of up to 127 characters. Spaces are not permitted in the string, but other special characters are allowed. In the global configuration mode, this command applies to all LDAP servers unless overridden in the specific LDAP host configuration.
Step 5	[no] ldap-server binddn Example: <code>apic1(config-host)# ldap-server binddn CN=ucsbind,OU=CiscoUsers,DC=sampledesign,DC=com</code>	Specifies the distinguished name (DN) for an LDAP database account that has read and search permissions for all objects under the base DN. This can be a string of up to 127 characters. Spaces are not permitted in the string, but other special characters are allowed.
Step 6	[no] ldap-server retries <i>count</i> Example: <code>apic1(config-host)# ldap-server retries 1</code>	Specifies how many times APIC transmits each LDAP request to the server before giving up. The range is 0 to 5. In the global configuration mode, this command applies to all LDAP servers unless overridden in the specific LDAP host configuration.
Step 7	[no] ldap-server timeout <i>seconds</i> Example: <code>apic1(config-host)# ldap-server timeout 30</code>	Specifies the number of seconds APIC waits for a reply to a LDAP request before retransmitting the request. In the global configuration mode, this command applies to all LDAP servers unless overridden in the specific LDAP host configuration.
Step 8	[no] ldap-server filter <i>filter-expression</i> Example: <code>apic1(config-host)# ldap-server filter sAMAccountName=\$userid</code>	Specifies a filter to filter the results of LDAP searches. The filter can contain a maximum of 63 characters. In the global configuration mode, this command applies to all LDAP servers unless overridden in the specific LDAP host configuration.

	Command or Action	Purpose
Step 9	[no] key <i>key-value</i> Example: <pre>apic1(config-host)# key Enter key: myLdAppassWoRd Enter key again: myLdAppassWoRd</pre>	Specifies the shared secret text string used between APIC and this LDAP server for authentication. The key can be up to 32 characters.
Step 10	[no] port <i>port-number</i> Example: <pre>apic1(config-host)# port 389</pre>	Specifies the LDAP server port for authentication.
Step 11	(Optional) [no] retries <i>count</i> Example: <pre>apic1(config-host)# retries 2</pre>	For this LDAP server, specifies how many times APIC transmits each LDAP request to the server before giving up. The range is 0 to 5. If no retry count is set, the global value is used.
Step 12	[no] enable-ssl Example: <pre>apic1(config-host)# enable-ssl</pre>	Enables an SSL connection with the LDAP provider.
Step 13	[no] ssl-validation-level [<i>permissive</i> <i>strict</i>] Example: <pre>apic1(config-host)# ssl-validation-level permissive</pre>	Sets the LDAP Server SSL Certificate validation level.
Step 14	(Optional) [no] timeout <i>seconds</i> Example: <pre>apic1(config-host)# timeout 3</pre>	For this LDAP server, specifies the number of seconds APIC waits for a reply to a LDAP request before retransmitting the request. If no timeout is set, the global value is used.
Step 15	exit Example: <pre>apic1(config-host)# exit</pre>	Returns to global configuration mode.
Step 16	show ldap-server Example: <pre>apic1(config)# show ldap-server</pre>	

Examples

This example shows how to configure LDAP server settings globally and on one LDAP server.

```
apic1# configure
apic1(config)# ldap-server retries 1
apic1(config)# ldap-server timeout 30
apic1(config)# ldap-server host 192.0.20.73
apic1(config-host)# retries 2
```

```

apic1(config-host)# timeout 3
apic1(config-host)# filter sAMAccountName=$userid
apic1(config-host)# key myLdAppassWoRd
apic1(config-host)# ssl-validation-level permissive
apic1(config-host)# enable-ssl
apic1(config-host)# port 389
apic1(config-host)# exit
apic1(config)# show ldap-server
timeout : 30
retries : 1
filter  : sAMAccountName=$userid

Total number of servers : 1

Hostname : 192.0.20.73
Port     : 389
Timeout  : 3
Retries  : 2
SSL      : yes
SSL Level : permissive
User     : test

```

Configuring the Password Policy

The password policy configuration in this topic set the password history and password change interval properties for all locally authenticated APIC users. You cannot specify different password policies for each locally authenticated user.

SUMMARY STEPS

1. **configure**
2. **[no] password change-count** *count*
3. **[no] password change-during-interval** {enable | disable}
4. **[no] password change-interval** *hours*
5. **[no] password no-change-interval** *hours*
6. **password expiration-warn-time**
7. **[no] password history-count** *count*
8. **[no] password pwd-strength-check**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	[no] password change-count <i>count</i> Example: apic1(config)# password change-count 5	Sets the number of password changes allowed within the change interval. The range is 0 to 10 changes.

	Command or Action	Purpose
Step 3	[no] password change-during-interval {enable disable} Example: <code>apicl(config)# password change-during-interval enable</code>	Enables or disables restricting the number of password changes a locally authenticated user can make within the change interval.
Step 4	[no] password change-interval <i>hours</i> Example: <code>apicl(config)# password change-interval 300</code>	When the change-during-interval is enabled, restricts the number of password changes a locally authenticated user can make within a given number of hours. The range is 1 to 745 hours.
Step 5	[no] password no-change-interval <i>hours</i> Example: <code>apicl(config)# password no-change-interval 60</code>	Sets a minimum period before which a user cannot change the password again. The range is 1 to 745 hours.
Step 6	password expiration-warn-time Example: <code>apicl(config)# password expiration-warn-time 5</code>	Sets a warning period before password expiration to display warning. The range is 0 to 30 days.
Step 7	[no] password history-count <i>count</i> Example: <code>apicl(config)# password history-count 10</code>	The password history count allows you to prevent locally authenticated users from reusing the same password over and over again. When this property is configured, APIC stores passwords that were previously used by locally authenticated users up to a maximum of 15 passwords. The passwords are stored in reverse chronological order with the most recent password first to ensure that the only the oldest password can be reused when the history count threshold is reached. A user must create and use the number of passwords configured in the password history count before being able to reuse one. For example, if you set the password history count to 8, a locally authenticated user cannot reuse the first password until after the ninth password has expired. By default, the password history is set to 0. This value disables the history count and allows users to reuse previous passwords at any time. If necessary, you can clear a user's password history using the clear-pwd-history command in the username configuration mode for that user.
Step 8	[no] password pwd-strength-check Example: <code>apicl(config)# password pwd-strength-check</code>	Enforces strong passwords for all users.

Examples

This example shows how to configure global password settings for locally authenticated users.

```

apic1# configure
apic1(config)# password change-count 5
apic1(config)# password change-during-interval enable
apic1(config)# password change-interval 300
apic1(config)# password no-change-interval 60
apic1(config)# password expiration-warn-time 5
apic1(config)# password history-count 10
apic1(config)# password pwd-strength-check

```

This example shows how to prevent the password from being changed within 48 hours after a locally authenticated user changes his or her password.

```

apic1# configure
apic1(config)# password change-during-interval disable
apic1(config)# password no-change-interval 48

```

This example shows how to allow the password to be changed a maximum of once within 24 hours after a locally authenticated user changes his or her password

```

apic1# configure
apic1(config)# password change-count 1
apic1(config)# password change-during-interval enable
apic1(config)# password change-interval 24

```

Configuring Users

Configuring a Locally Authenticated User

SUMMARY STEPS

1. **configure**
2. **username** *{name | admin}*
3. **[no] first-name** *first*
4. **[no] last-name** *last*
5. **[no] email** *email-address*
6. **[no] phone** *phone-number*
7. **[no] account-status** *{active | inactive | status}*
8. **clear-pwd-history**
9. **[no] expires**
10. **expiration** *date-time*
11. **password** *password*
12. **[no] pwd-lifetime** *days*
13. **[no] domain** *{all | common | mgmt | domain-name}*
14. **[no] role** *role*
15. **[no] priv-type** *{readPriv | writePriv}*

- 16. `exit`
- 17. `exit`
- 18. `show username name`

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	username <i>{name admin}</i> Example: <code>apic1(config)# username user5</code>	Creates a locally-authenticated user account or configures an existing user. The name can be a maximum of 28 characters.
Step 3	[no] first-name <i>first</i> Example: <code>apic1(config-username)# first-name George</code>	Sets the first name of this user.
Step 4	[no] last-name <i>last</i> Example: <code>apic1(config-username)# last-name Washington</code>	Sets the last name of this user.
Step 5	[no] email <i>email-address</i> Example: <code>apic1(config-username)# email gwashington@exampleCorp.com</code>	Sets the email address of this user.
Step 6	[no] phone <i>phone-number</i> Example: <code>apic1(config-username)# phone 14085551212</code>	Sets the phone number of this user.
Step 7	[no] account-status <i>{active inactive status}</i> Example: <code>apic1(config-username)# account-status active</code>	Activates or deactivates this user account.
Step 8	clear-pwd-history Example: <code>apic1(config-username)# clear-pwd-history</code>	Clears the user's password history list and allows this user to reuse previous passwords.
Step 9	[no] expires Example: <code>apic1(config-username)# expires</code>	Enables expiration of this user account at the date and time configured by the expiration command.
Step 10	expiration <i>date-time</i> Example:	Sets an expiration date and time for this user account. The format is UTC Date format

	Command or Action	Purpose
	<code>apic1(config-username) # expiration 2017-12-31T23:59+08:00</code>	(YYYY-MM-DDThh:mmTZD). You must also enable expiration by configuring the expires command.
Step 11	password <i>password</i> Example: <code>apic1(config-username) # password c1\c0123</code>	Sets the user password. Note Special characters such as '\$' or '!' should be escaped with a backslash ('\') in this command to avoid misinterpretation by Bash. The escape backslash is necessary only when setting the password in this command; the user does not enter the backslash when logging in.
Step 12	[no] pwd-lifetime <i>days</i> Example: <code>apic1(config-username) # pwd-lifetime 90</code>	Sets the lifetime of the user password. The range is 0 to 3650 days.
Step 13	[no] domain {all common mgmt <i>domain-name</i> } Example: <code>apic1(config-username) # domain mySecDomain</code>	Specifies or creates the AAA domain to which this user belongs.
Step 14	[no] role <i>role</i> Example: <code>apic1(config-domain) # role tenant-admin</code>	Creates the AAA domain role to set privilege bitmask of a user domain.
Step 15	[no] priv-type {readPriv writePriv} Example: <code>apic1(config-role) # priv-type writePriv</code>	Creates the AAA domain role to set privilege bitmask of a user domain.
Step 16	exit Example: <code>apic1(config-role) # exit</code>	Returns to domain configuration mode.
Step 17	exit Example: <code>apic1(config-domain) # exit</code>	Returns to username configuration mode.
Step 18	show username <i>name</i> Example: <code>apic1(config-username) # show username user5</code>	Displays configuration details about this user.

Examples

This example shows how to configure a local user.

```
apic1# configure terminal
apic1(config)# username user5
```

```

apicl(config-username)# first-name George
apicl(config-username)# last-name Washington
apicl(config-username)# email gwashington@exampleCorp.com
apicl(config-username)# phone 14085551212
apicl(config-username)# account-status active
apicl(config-username)# domain mySecDomain
apicl(config-username)# clear-pwd-history
apicl(config-username)# expires
apicl(config-username)# expiration 2017-12-31T23:59+08:00
apicl(config-username)# password c1$c0123
apicl(config-username)# pwd-lifetime 90
apicl(config-username)# domain mySecDomain
apicl(config-domain)# role tenant-admin
apicl(config-role)# priv-type writePriv
apicl(config-role)# exit
apicl(config-domain)# exit
apicl(config-username)# show username user5
UserName                : user5
First-Name              : George
Last-Name               : Washington
Email                  : gwashington@exampleCorp.com
Account Status         : active
Password strength check : yes

```

What to do next

To configure an SSH key or certificate for the local user, see "Configuring Certificates and SSH-Keys."

Configuring a Certificate and SSH-Key for a Local User

This topic describes how to configure a certificate or an SSH key so that a local user can log in without being prompted for a password.

SUMMARY STEPS

1. **configure**
2. **username** *{name | admin}*
3. **[no] certificate** *certificate-name*
4. **data** *certificate-data*
5. **exit**
6. **[no] ssh-key** *ssh-key-name*
7. **data** *key-data*
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.

	Command or Action	Purpose
Step 2	username { <i>name</i> admin } Example: <code>apic1(config)# username user5</code>	Creates a locally-authenticated user account or configures an existing user. The name can be a maximum of 28 characters.
Step 3	[no] certificate <i>certificate-name</i> Example: <code>apic1(config-username)# certificate myCertificate</code>	Enters certificate configuration mode.
Step 4	data <i>certificate-data</i> Example: <code>apic1(config-certificate)# data -----BEGIN CERTIFICATE-----MIIC4j.....</code>	Sets PEM-encoded certificate.
Step 5	exit Example: <code>apic1(config-certificate)# exit</code>	Returns to username configuration mode.
Step 6	[no] ssh-key <i>ssh-key-name</i> Example: <code>apic1(config-username)# ssh-key mySSHkey</code>	Sets an SSH key to log in using the SSH client without being prompted for a password.
Step 7	data <i>key-data</i> Example: <code>apic1(config-ssh-key)# data AAAAB3NzaC1yc2EAA.....</code>	Sets the SSH key. The key can be up to 64 characters.
Step 8	exit Example: <code>apic1(config-ssh-key)# exit</code>	Returns to username configuration mode.

Examples

This example shows how to configure an SSH key and a certificate for a local user.

```
apic1# configure terminal
apic1(config)# username user5
apic1(config-username)# certificate myCertificate
apic1(config-certificate)# data -----BEGIN CERTIFICATE-----MIIC4j.....
apic1(config-certificate)# exit
apic1(config-username)# ssh-key mySSHkey
apic1(config-ssh-key)# data AAAAB3NzaC1yc2EAA...
apic1(config-ssh-key)# exit
```

Configuring Public Key Infrastructure

Configuring a Certificate Authority and Chain of Trust

Certificate authorities (CAs) manage certificate requests and issue certificates to participating entities such as hosts, network devices, or users. APIC locally stores the self-signed root certificate of the trusted CA (or certificate chain for a subordinate CA). The stored information about a trusted CA is called the trustpoint and the CA itself is called a trustpoint CA.

SUMMARY STEPS

1. **configure**
2. **[no] crypto ca *trustpoint-name***
3. **[no] cert-chain *pem-data***

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] crypto ca <i>trustpoint-name</i> Example: <code>apic1(config)# crypto ca myCA</code>	Enters configuration mode for the specified trustpoint certificate authority (CA).
Step 3	[no] cert-chain <i>pem-data</i> Example: <code>apic1(config-ca)# cert-chain -----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....</code>	Stores the certificate chain in PEM format. Enter the entire chain of trust from the trustpoint to a trusted root authority.

Examples

This example shows how to configure a CA.

```
apic1# configure

apic1(config)# crypto ca myCA
apic1(config-ca)# cert-chain -----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....
```

Configuring Keys and a Keyring

You can obtain an identity certificate for APIC by generating an RSA key pair and associating the key pair with a trustpoint CA where APIC intends to enroll. The RSA keys are stored by APIC in a crypto keyring.

The APIC software allows you to generate an RSA key pair with a configurable key size (or modulus). The default key size is 512. You can also configure an RSA key-pair label. The default key label is the device fully qualified domain name (FQDN).

SUMMARY STEPS

1. **configure**
2. **[no] crypto keyring {default | *keyring-name*}**
3. **regen**
4. **[no] cert *certificate-data***
5. **[no] tp *certificate-name***
6. **[no] key *key-data***
7. **[no] modulus {mod512 | mod1024 | mod1536 | mod2048}**
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] crypto keyring {default <i>keyring-name</i>} Example: <code>apic1(config)# crypto keyring myKeyring</code>	Creates or configures a keyring to hold an SSL certificate.
Step 3	regen Example: <code>apic1(config-keyring)# regen</code>	Forces regeneration of the RSA key pair.
Step 4	[no] cert <i>certificate-data</i> Example: <code>apic1(config-keyring)# cert "-----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....</code>	Imports a certificate containing a public key and signed information. The certificate data must be enclosed in quotes.
Step 5	[no] tp <i>certificate-name</i> Example: <code>apic1(config-keyring)# tp myCertificate</code>	Sets a third-party certificate from a trusted source for device identity.
Step 6	[no] key <i>key-data</i> Example: <code>apic1(config-keyring)# key XXXXXXXXXXXXXXXXXXXXXXXX</code>	Creates the private key of the certificate.
Step 7	[no] modulus {mod512 mod1024 mod1536 mod2048} Example: <code>apic1(config-keyring)# modulus mod1024</code>	Sets the length of the encryption keys.

	Command or Action	Purpose
Step 8	exit Example: <code>apicl(config-keyring)# exit</code>	Returns to global configuration mode.

Examples

This example shows how to configure a keyring.

```

apicl# configure
apicl(config)# crypto keyring myKeyring
apicl(config-keyring)# cert "-----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....
apicl(config-keyring)# tp myCertificate
apicl(config-keyring)# key XXXXXXXXXXXXXXXXXXXXXXXXXX
apicl(config-keyring)# modulus mod1024
apicl(config-keyring)# exit

```

Generating a Certificate Signing Request

A certificate signing request (CSR) is a message that an applicant sends to a CA in order to apply for a digital identity certificate. Before a CSR is created, the applicant first generates a key pair, which keeps the private key secret. The CSR contains information that identifies the applicant, such as the public key generated by the applicant. The corresponding private key is not included in the CSR, but is used to digitally sign the entire request.

Before you begin

Before generating a certificate signing request (CSR), you must configure a trustpoint certificate authority (CA) and generate a key pair.

SUMMARY STEPS

1. **configure**
2. **[no] crypto keyring {default | *keyring-name*}**
3. **csr**
4. **subj-name *name***
5. **[no] cert *certificate-data***
6. **password**
7. **org-name**
8. **org-unit-name**
9. **email**
10. **locality *city-name***
11. **state *state***
12. **country *country-code***
13. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	[no] crypto keyring {default <i>keyring-name</i>} Example: apicl(config)# crypto keyring default	Creates or configures a keyring to hold an SSL certificate.
Step 3	csr Example: apicl(config-keyring)# csr	Creates a certificate signing request for this keyring.
Step 4	subj-name <i>name</i> Example: apicl(config-csr)# subj-name www.exampleCorp.com	Sets the fully qualified domain name or distinguished name of the requesting device. The name can be up to 64 characters.
Step 5	[no] cert <i>certificate-data</i> Example: apicl(config-csr)# cert "-----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....	Imports a certificate containing a public key and signed information. The certificate data must be enclosed in quotes.
Step 6	password Example: apicl(config-csr)# password Enter password: c1\$c0123 Enter password again: c1\$c0123	Sets the new password.
Step 7	org-name Example: apicl(config-csr)# org-name ExampleCorp	Sets the full legal name of the organization.
Step 8	org-unit-name Example: apicl(config-csr)# org-unit-name Sales	Sets the department or unit name within the organization.
Step 9	email Example: apicl(config-csr)# email admin@exampleCorp.com	Sets the email address of the organization contact person.
Step 10	locality <i>city-name</i> Example: apicl(config-csr)# locality SanJose	Sets the city or town of the organization.

	Command or Action	Purpose
Step 11	state <i>state</i> Example: <code>apicl(config-csr)# state CA</code>	Sets the state or province in which the organization is located.
Step 12	country <i>country-code</i> Example: <code>apicl(config-csr)# country US</code>	Sets the two-letter ISO code for the country where the organization is located.
Step 13	exit Example: <code>apicl(config-csr)# exit</code>	Returns to keyring configuration mode.

Examples

This example shows how to generate a certificate signing request (CSR).

```

apicl# configure
apicl(config)# crypto keyring default
apicl(config-keyring)# csr
apicl(config-csr)# subj-name www.exampleCorp.com
apicl(config-csr)# cert "-----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....
apicl(config-csr)# pwd c1$c0123
apicl(config-csr)# org-name ExampleCorp
apicl(config-csr)# org-unit-name Sales
apicl(config-csr)# email admin@exampleCorp.com
apicl(config-csr)# locality SanJose
apicl(config-csr)# state CA
apicl(config-csr)# country US
apicl(config-csr)# exit

```

What to do next

Submit the CSR to a CA.

Configuring Webtokens

SUMMARY STEPS

1. **configure**
2. **[no] crypto webtoken**
3. **[no] max-validity-period** *hours*
4. **[no] session-record-flags** *csv-list*
5. **[no] ui-idle-timeout-seconds** *seconds*
6. **[no] webtoken-timeout-seconds** *seconds*
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] crypto webtoken Example: <code>apic1(config)# crypto webtoken</code>	
Step 3	[no] max-validity-period <i>hours</i> Example: <code>apic1(config-webtoken)# max-validity-period 10</code>	Sets the maximum validity period for a webtoken. The range is 4 to 24 hours.
Step 4	[no] session-record-flags <i>csv-list</i> Example: <code>apic1(config-webtoken)# session-record-flags login,refresh</code>	Enables or disables refresh in the session records. The session record flags are specified as a comma-separated value list of one or more of the following flags: login , logout , and refresh .
Step 5	[no] ui-idle-timeout-seconds <i>seconds</i> Example: <code>apic1(config-webtoken)# ui-idle-timeout-seconds 120</code>	Sets the maximum GUI idle duration before requiring login refresh. The range is 60 to 65525 seconds.
Step 6	[no] webtoken-timeout-seconds <i>seconds</i> Example: <code>apic1(config-webtoken)# webtoken-timeout-seconds 1200</code>	Sets the webtoken timeout interval. The range is 600 to 9600 seconds.
Step 7	exit Example: <code>apic1(config-webtoken)# exit</code>	

Examples

This example shows how to configure a webtoken.

```
apic1# configure
apic1(config)# crypto webtoken
apic1(config-webtoken)# max-validity-period 10
apic1(config-webtoken)# session-record-flags login,refresh
apic1(config-webtoken)# ui-idle-timeout-seconds 120
apic1(config-webtoken)# webtoken-timeout-seconds 1200
```

Configuring Communication Policies

Configuring the HTTP Policy

SUMMARY STEPS

1. **configure**
2. **[no] comm-policy {default | *policy-name*}**
3. **http**
4. **[no] admin-state-enable**
5. **[no] allow-origin *url***
6. **[no] port *port-number***
7. **[no] redirect**
8. **[no] request-status-count *count***
9. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	[no] comm-policy {default <i>policy-name</i>} Example: apicl(config)# comm-policy myCommPolicy	Enters communication policy configuration mode.
Step 3	http Example: apicl(config-comm-policy)# http	Enters HTTP policy configuration mode.
Step 4	[no] admin-state-enable Example: apicl(config-http)# admin-state-enable	Enables HTTP communication service.
Step 5	[no] allow-origin <i>url</i> Example: apicl(config-http)# allow-origin www.example.com	Specifies the URL to return in the Access-Control-Allow-Origin HTTP header.
Step 6	[no] port <i>port-number</i> Example: apicl(config-http)# port 8080	Sets the port used for HTTP communication service.

	Command or Action	Purpose
Step 7	[no] redirect Example: <code>apic1(config-http)# no redirect</code>	Enables HTTP redirection.
Step 8	[no] request-status-count <i>count</i> Example: <code>apic1(config-http)# request-status-count 512</code>	Sets the maximum count of HTTP requests to track. The range is 0 to 10240.
Step 9	exit Example: <code>apic1(config-http)# exit</code>	Returns to communications policy configuration mode.

Examples

This example shows how to configure HTTP service.

```
apic1# configure
apic1(config)# comm-policy myCommPolicy
apic1(config-comm-policy)# http
apic1(config-http)# admin-state-enable
apic1(config-http)# allow-origin www.example.com
apic1(config-http)# port 8080
apic1(config-http)# no redirect
apic1(config-http)# request-status-count 512
apic1(config-http)# exit
```

Configuring the HTTPS Policy

SUMMARY STEPS

1. **configure**
2. **[no] comm-policy {default | *policy-name*}**
3. **https**
4. **[no] admin-state-enable**
5. **[no] port *port-number***
6. **[no] request-status-count *count***
7. **[no] ssl-protocols {TLSv1 | TLSv1.1 | TLSv1.2}**
8. **[no] use-keyring *keyring-name***
9. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	[no] comm-policy {default <i>policy-name</i>} Example: apicl(config)# comm-policy myCommPolicy	Enters communication policy configuration mode.
Step 3	https Example: apicl(config-comm-policy)# https	Enters HTTPS policy configuration mode.
Step 4	[no] admin-state-enable Example: apicl(config-https)# admin-state-enable	Enables HTTPS communication service.
Step 5	[no] port <i>port-number</i> Example: apicl(config-https)# port 443	Sets the port used for HTTPS communication service.
Step 6	[no] request-status-count <i>count</i> Example: apicl(config-https)# request-status-count 512	Sets the maximum count of HTTPS requests to track. The range is 0 to 10240.
Step 7	[no] ssl-protocols {TLSv1 TLSv1.1 TLSv1.2} Example: apicl(config-https)# ssl-protocols TLSv1.1,TLSv1.2	Specifies in a comma-separated list the SSL protocols that are supported. The options are TLSv1 , TLSv1.1 , and TLSv1.2 .
Step 8	[no] use-keyring <i>keyring-name</i> Example: apicl(config-https)# use-keyring myKeyRing	Specifies a keyring to use for the HTTPS server SSL certificate.
Step 9	exit Example: apicl(config-https)# exit	Returns to communications policy configuration mode.

Examples

This example shows how to configure HTTPS service.

```

apicl# configure
apicl(config)# comm-policy myCommPolicy
apicl(config-comm-policy)# https

```

```

apicl(config-https)# admin-state-enable
apicl(config-https)# port 443
apicl(config-https)# request-status-count 512
apicl(config-https)# ssl-protocols TLSv1.1,TLSv1.2
apicl(config-https)# use-keyring myKeyRing
apicl(config-https)# exit

```

Configuring the SSH Policy

SUMMARY STEPS

1. **configure**
2. **[no] comm-policy {default | *policy-name*}**
3. **ssh-service**
4. **[no] admin-state-enable**
5. **[no] port *port-number***
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	[no] comm-policy {default <i>policy-name</i>} Example: apicl(config)# comm-policy myCommPolicy	Enters communication policy configuration mode.
Step 3	ssh-service Example: apicl(comm-policy)# ssh-service	Enters SSH policy configuration mode.
Step 4	[no] admin-state-enable Example: apicl(config-ssh-service)# admin-state-enable	Enables HTTP communication service.
Step 5	[no] port <i>port-number</i> Example: apicl(config-ssh-service)# port 22	Sets the port used for SSH communication service.
Step 6	exit Example: apicl(config-ssh-service)# exit	Returns to communications policy configuration mode.

Examples

This example shows how to configure SSH service.

```
apicl# configure
apicl(config)# comm-policy myCommPolicy
apicl(config-comm-policy)# ssh-service
apicl(config-ssh-service)# admin-state-enable
apicl(config-ssh-service)# port 22
apicl(config-ssh-service)# exit
```

Configuring the Telnet Policy

Before you begin

To allow telnet communications, you must configure an out-of-band contract allowing telnet traffic, which is normally on TCP and UDP ports 23.

SUMMARY STEPS

1. **configure**
2. **[no] comm-policy {default | *policy-name*}**
3. **telnet**
4. **[no] admin-state-enable**
5. **[no] port *port-number***
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	[no] comm-policy {default <i>policy-name</i>} Example: apicl(config)# comm-policy myCommPolicy	Enters communication policy configuration mode.
Step 3	telnet Example: apicl(config-comm-policy)# telnet	Enters Telnet policy configuration mode.
Step 4	[no] admin-state-enable Example: apicl(config-telnet)# admin-state-enable	Enables Telnet communication service.

	Command or Action	Purpose
Step 5	[no] port <i>port-number</i> Example: <code>apic1(config-telnet) # port 23</code>	Sets the port used for Telnet communication service.
Step 6	exit Example: <code>apic1(config-telnet) # exit</code>	Returns to communications policy configuration mode.

Examples

This example shows how to configure Telnet service.

```
apic1# configure
apic1(config)# comm-policy myCommPolicy
apic1(config-comm-policy)# telnet
apic1(config-telnet)# admin-state-enable
apic1(config-telnet)# port 23
apic1(config-telnet)# exit
```

Configuring AES Encryption

Beginning with Cisco APIC 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. For a list of secure properties, see "Appendix K: Secure Properties" in *Cisco Application Centric Infrastructure Fundamentals*.

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 whether the same passphrase is 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.

SUMMARY STEPS

1. **configure**
2. **crypto aes**
3. (Optional) **clear-encryption-key**
4. **passphrase**
5. **[no] encryption**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters configuration mode.
Step 2	crypto aes Example: apicl(config)# crypto aes	Enters AES configuration mode.
Step 3	(Optional) clear-encryption-key Example: apicl(config-aes)# clear-encryption-key	Deletes any existing AES encryption key.
Step 4	passphrase Example: apicl(config-aes)# passphrase Enter passphrase: "This is my passphrase" Enter passphrase again: "This is my passphrase"	Specifies the AES encryption passphrase. The passphrase can be 16 to 32 characters and must be enclosed in quotes. For increased security, entering the passphrase is interactive.
Step 5	[no] encryption Example: apicl(config-aes)# encryption	Enables (or disables) AES encryption.

Examples

This example shows how to enable AES encryption and configure a passphrase.

```

apicl# configure
apicl(config)# crypto aes
apicl(config-aes)# clear-encryption-key
apicl(config-aes)# passphrase "This is my passphrase"
apicl(config-aes)# encryption

```

Configuring 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 Cisco APIC 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). An administrator with fabric-wide access rights must enable strict mode.

Permissive Mode (default) operates as follows:

- Allows an existing fabric to operate normally even though one or more switches have an invalid certificate.
- Does not enforce serial number based authorization.
- Allows auto-discovered controllers and switches to join the fabric without enforcing serial number authorization.

Strict Mode operates as follows:

- Only switches with a valid Cisco serial number and SSL certificate are allowed.
- Enforces serial number based authorization.
- Requires an administrator to manually authorize controllers and switches to join the fabric.

SUMMARY STEPS

1. **configure**
2. **system fabric-security-mode {permissive | strict}**
3. **system controller-id *controller-id* {approve | reject}**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters configuration mode.
Step 2	system fabric-security-mode {permissive strict} Example: <code>apic1(config)# system fabric-security-mode strict</code>	Specifies the fabric security mode.
Step 3	system controller-id <i>controller-id</i> {approve reject} Example: <code>apic1(config)# system controller-id FCH1750V025 approve</code>	In strict mode, approves or rejects a controller to join the fabric.

Examples

This example shows how to change the fabric security mode to strict.

```
apic1# configure
apic1(config)# system fabric-security-mode strict
```

This example shows how to approve a controller to join the fabric when strict mode is configured.

```
apic1# configure
apic1(config)# system controller-id FCH1750V025 approve
```

Configuring COOP Authentication

About COOP Authentication

Council of Oracles Protocol (COOP) is used to communicate the mapping information (location and identity) to the spine proxy. A leaf switch will forward endpoint address information to a spine using ZeroMQ (Zero Message Queue or ZMQ). COOP running on the spine nodes ensures that all spine nodes maintain a consistent copy of end point address and location information and additionally maintains the distributed hash table (DHT) repository of endpoint identity to location mapping database.

Without COOP authentication, it is possible for users to send arbitrary COOP messages, which would be acted on by the fabric nodes. Cisco APIC Release 2.0 adds an MD5 TCP option to provide authentication and integrity protection to the ZMQ TCP transportation. Two authentication modes are supported:

- **Compatible** - COOP accepts both MD5 authenticated and non-authenticated ZMQ connections for message transportation. COOP data path communication gives high priority to transport via secured connections.
- **Strict** - COOP allows MD5 authenticated ZMQ connections only.

Changing the configuration of the COOP authentication type has the following effects:

- When the configuration changes from compatible to strict mode, all non-authenticated ZMQ connections are disconnected.
- When the configuration changes from strict to compatible mode, COOP immediately accepts both authenticated and non-authenticated ZMQ connections.

Configuring COOP Authentication

SUMMARY STEPS

1. **configure**
2. **coop-fabric**
3. **authentication type {compatible | strict}**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	coop-fabric Example: <code>apic1(config)# coop-fabric</code>	Enters COOP fabric configuration mode.

	Command or Action	Purpose
Step 3	authentication type {compatible strict} Example: <pre>apic1(config-coop-fabric)# authentication type compatible</pre>	Configures the COOP authentication type as one of the following: • compatible - COOP allows MD5 authenticated and non-authenticated ZMQ connections. • strict - allows MD5 authenticated ZMQ connections only.

Example

This example shows how to configure COOP authentication in compatible mode:

```
apic1# configure
apic1(config)# coop-fabric
apic1(config-coop-fabric)# authentication type compatible
```

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

Follow these guidelines and limitations:

- When FIPS is enabled, it is applied across Cisco APIC.
- When performing a Cisco APIC software downgrade, you must disable FIPS first.
- Make your passwords a minimum of eight characters in length.
- Disable Telnet. Users should log in using SSH only.
- Delete all SSH Server RSA1 keypairs.
- Disable remote authentication through RADIUS/TACACS+. Only local and LDAP users can be authenticated.
- 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.
- Starting with release 2.3(1x), FIPS can be configured at the switch level.
- Starting with release 3.1(1x), 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 NX-OS Style CLI

When FIPS is enabled, it is applied across Cisco APIC.

SUMMARY STEPS

1. **configure**
2. **fips mode enable**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apicl# configure</code>	Enters configuration mode.
Step 2	fips mode enable Example: <code>apicl(config)# fips mode enable</code>	Enables FIP. The no fips mode enable command disables FIPS. You must reboot to complete the configuration. Anytime you change the mode, you must reboot to complete the configuration.

Configuring Control Plane Policing

Information About CoPP

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 ACI Leaf/Spine 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 an ACI Leaf/Spine CPU or CPU itself.

The supervisor module of ACI Leaf/Spine 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 ACI Leaf/Spine 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 ACI Leaf/Spine 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 ACI Leaf/Spines 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 ACI Leaf/Spines 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. Another example is a packet with IP options set.

- **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 ACI Leaf/Spine supervisor module receives these packets.

Classification for CoPP:

For effective protection, the ACI Leaf/Spine 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.

Rate Controlling Mechanisms:

Once the packets are classified, the ACI Leaf/Spine 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/Spine 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.

- 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 Cisco NX-OS CLI

Step 1 Configure a CoPP leaf profile:

Example:

```
# configure copp Leaf Profile
apic1(config)# policy-map type control-plane-leaf leafProfile
apic1(config-pmap-copp-leaf)# profile-type custom
apic1(config-pmap-copp-leaf)# set arpRate 786
# create a policy group to be applied on leaves
apic1(config)# template leaf-policy-group coppForLeaves
apic1(config-leaf-policy-group)# copp-aggr leafProfile
apic1(config-leaf-policy-group)# exit
# apply the leaves policy group on leaves
apic1(config)# leaf-profile applyCopp
apic1(config-leaf-profile)# leaf-group applyCopp
apic1(config-leaf-group)# leaf 101-102
apic1(config-leaf-group)# leaf-policy-group coppForLeaves
```

Step 2 Configure a CoPP Spine profile:

Example:

```
# configure copp Spine Profile
apic1(config)# policy-map type control-plane-spine spineProfile
apic1(config-pmap-copp-spine)# profile-type custom
apic1(config-pmap-copp-spine)# set arpRate 786
# create a policy group to be applied on spines
apic1(config)# template leaf-policy-group coppForSpines
apic1(config-spine-policy-group)# copp-aggr spineProfile
apic1(config-spine-policy-group)# exit
# apply the spine policy group on spines
apic1(config)# spine-profile applyCopp
apic1(config-spine-profile)# spine-group applyCopp
apic1(config-spine-group)# spine 201-202
apic1(config-spine-group)# spine-policy-group coppForSpines
```

Configuring Per Interface Per Protocol CoPP Policy Using the NX-OS Style CLI

Step 1 Define the CoPP class map and policy map:

Example:

```
(config)# policy-map type control-plane-if <name>
(config-pmap-copp)# protocol bgp bps <value>
(config-pmap-copp)# protocol ospf bps <value>
```

Step 2 Applying the configuration to an interface on the leaf:

Example:

```
(config)# leaf 101
(config-leaf)# int eth 1/10
(config-leaf-if)# service-policy type control-plane-if output<name>
```

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 Using the NX-OS CLI

Before you begin

- The tenant and Bridge Domain configured.

Step 1 **configure**

Enters configuration mode.

Example:

```
apic1# configure
```

Step 2 **Configure FHS policy.****Example:**

```
apic1(config)# tenant coke
apic1(config-tenant)# first-hop-security
apic1(config-tenant-fhs)# security-policy poll
apic1(config-tenant-fhs-secpol)#
apic1(config-tenant-fhs-secpol)# ip-inspection-admin-status enabled-both
apic1(config-tenant-fhs-secpol)# source-guard-admin-status enabled-both
apic1(config-tenant-fhs-secpol)# router-advertisement-guard-admin-status enabled
apic1(config-tenant-fhs-secpol)# router-advertisement-guard
apic1(config-tenant-fhs-raguard)#
apic1(config-tenant-fhs-raguard)# managed-config-check
apic1(config-tenant-fhs-raguard)# managed-config-flag
apic1(config-tenant-fhs-raguard)# other-config-check
apic1(config-tenant-fhs-raguard)# other-config-flag
apic1(config-tenant-fhs-raguard)# maximum-router-preference low
apic1(config-tenant-fhs-raguard)# minimum-hop-limit 10
apic1(config-tenant-fhs-raguard)# maximum-hop-limit 100
apic1(config-tenant-fhs-raguard)# exit
apic1(config-tenant-fhs-secpol)# exit
apic1(config-tenant-fhs)# trust-control tcpoll
apic1(config-tenant-fhs-trustctrl)# arp
apic1(config-tenant-fhs-trustctrl)# dhcpv4-server
apic1(config-tenant-fhs-trustctrl)# dhcpv6-server
apic1(config-tenant-fhs-trustctrl)# ipv6-router
apic1(config-tenant-fhs-trustctrl)# router-advertisement
apic1(config-tenant-fhs-trustctrl)# neighbor-discovery
apic1(config-tenant-fhs-trustctrl)# exit
apic1(config-tenant-fhs)# exit
apic1(config-tenant)# bridge-domain bd1
apic1(config-tenant-bd)# first-hop-security security-policy poll
apic1(config-tenant-bd)# exit
apic1(config-tenant)# application ap1
apic1(config-tenant-app)# epq epq1
apic1(config-tenant-app-epq)# first-hop-security trust-control tcpoll
```

Step 3 **Show FHS configuration example:****Example:**

```
leaf4# show fhs bt all
```

Legend:

TR	: trusted-access	UNRES	: unresolved	Age	: Age since
creation					
UNTR	: untrusted-access	UNDTR	: undetermined-trust	CRTNG	: creating
UNKNW	: unknown	TENTV	: tentative	INV	: invalid
NDP	: Neighbor Discovery Protocol	STA	: static-authenticated	REACH	: reachable
INCPM	: incomplete	VERFY	: verify	INTF	: Interface


```

TimeLeft : Remaining time since last refresh      LM      : lla-mac-match      DHCP   :
dhcp-assigned

```

```

EPG-Mode:
  U : unknown   M : mac     V : vlan    I : ip

```

```

BD-VNID      BD-Vlan      BD-Name
15630220     3                t0:bd200

```

Origin	IP	MAC	INTF	EPG(sclass) (mode)	Trust-lvl	State
Age	TimeLeft					
ARP	192.0.200.12	D0:72:DC:A0:3D:4F	eth1/1	ep300(49154) (V)	LM,TR	STALE
00:04:49	18:08:13					
ARP	172.29.205.232	D0:72:DC:A0:3D:4F	eth1/1	ep300(49154) (V)	LM,TR	STALE
00:03:55	18:08:21					
ARP	192.0.200.21	D0:72:DC:A0:3D:4F	eth1/1	ep300(49154) (V)	LM,TR	REACH
00:03:36	00:00:02					
LOCAL	192.0.200.1	00:22:BD:F8:19:FF	vlan3	LOCAL(16387) (I)	STA	REACH
04:49:41	N/A					
LOCAL	fe80::200	00:22:BD:F8:19:FF	vlan3	LOCAL(16387) (I)	STA	REACH
04:49:40	N/A					
LOCAL	2001:0:0:200::1	00:22:BD:F8:19:FF	vlan3	LOCAL(16387) (I)	STA	REACH
04:49:39	N/A					

The trust levels are:

- **TR**—Trusted. Displayed when the endpoint is learned from an EPG where the trust configuration is enabled.
- **UNTR**—Untrusted. Displayed when the endpoint is learned from an EPG where the trust configuration is not enabled.
- **UNDTR**—Undetermined. Displayed in the case of a DHCP relay topology where the DHCP server bridge domain (BD) is on a remote leaf and the DHCP clients are on a local leaf. In this situation, the local leaf will not know whether the DHCP server BD has trust DHCP enabled.

Step 4 Show violations with the different types and reasons example:

Example:

```
leaf4# show fhs violations all
```

```

Violation-Type:
  POL : policy      THR : address-theft-remote
  ROLE : role       TH  : address-theft
  INT  : internal

```

```

Violation-Reason:
  IP-MAC-TH : ip-mac-theft      OCFG_CHK : ra-other-cfg-check-fail  ANC-COL
  : anchor-collision
  PRF-LVL-CHK : ra-rtr-pref-level-check-fail  INT-ERR  : internal-error          TRUST-CHK
  : trust-check-fail
  SRV-ROL-CHK : srv-role-check-fail          ST-EP-COL : static-ep-collision        LCL-EP-COL
  : local-ep-collision
  MAC-TH      : mac-theft        EP-LIM    : ep-limit-reached           MCFG-CHK
  : ra-managed-cfg-check-fail
  HOP-LMT-CHK : ra-hoplimit-check-fail      MOV-COL   : competing-move-collision        RTR-ROL-CHK
  : rtr-role-check-fail
  IP-TH       : ip-theft

```

```

EPG-Mode:
  U : unknown   M : mac     V : vlan    I : ip

```

BD-VNID	BD-Vlan	BD-Name				
15630220	3	t0:bd200				

Type	Last-Reason	Proto	IP	MAC	Port	EPG(sclass) (mode) Count

THR	IP-TH	ARP	192.0.200.21	D0:72:DC:A0:3D:4F	tunnel5	epg300(49154) (V) 21

Table Count: 1						

Table Count: 1

Step 5 Show FHS configuration:

Example:

```
swtb23-ifc1# show tenant t0 bridge-domain bd200 first-hop-security binding-table
```

Pod/Node State	Type	Family	IP Address	MAC Address	Interface	Level

1/102 reach	local	ipv4	192.0.200.1	00:22:BD:F8:19:FF	vlan3	static- authenticated
able 1/102 reach	local	ipv6	fe80::200	00:22:BD:F8:19:FF	vlan3	static- authenticated
able 1/102 reach	local	ipv6	2001:0:0:200::1	00:22:BD:F8:19:FF	vlan3	static- authenticated
able 1/101 stale	arp	ipv4	192.0.200.23	D0:72:DC:A0:02:61	eth1/2	lla-mac-match ,untrusted-access static-
1/101 reach	local	ipv4	192.0.200.1	00:22:BD:F8:19:FF	vlan3	authenticated
able 1/101 reach	nd	ipv6	fe80::d272:dcff:fea0:261	D0:72:DC:A0:02:61	eth1/2	lla-mac-match ,untrusted-access
able 1/101 stale	nd	ipv6	2001:0:0:200::20	D0:72:DC:A0:02:61	eth1/2	lla-mac-match ,untrusted-access
1/101 stale	nd	ipv6	2001::200:d272:dcff:fea0:261	D0:72:DC:A0:02:61	eth1/2	lla-mac-match ,untrusted-access
1/101 reach	local	ipv6	fe80::200	00:22:BD:F8:19:FF	vlan3	static- authenticated
able 1/101 reach	local	ipv6	2001:0:0:200::1	00:22:BD:F8:19:FF	vlan3	static- authenticated
able 1/103 reach	local	ipv4	192.0.200.1	00:22:BD:F8:19:FF	vlan4	static-

able						authenticated
1/103	local	ipv6	fe80::200	00:22:BD:F8:19:FF	vlan4	static-
reach						authenticated
able						authenticated
1/103	local	ipv6	2001:0:0:200::1	00:22:BD:F8:19:FF	vlan4	static-
reach						authenticated
able						authenticated
1/104	arp	ipv4	192.0.200.10	F8:72:EA:AD:C4:7C	eth1/1	lla-mac-match
stale						,trusted-access
1/104	arp	ipv4	172.29.207.222	D0:72:DC:A0:3D:4C	eth1/1	lla-mac-match
stale						,trusted-access
1/104	local	ipv4	192.0.200.1	00:22:BD:F8:19:FF	vlan4	static-
reach						authenticated
able						authenticated
1/104	nd	ipv6	fe80::fa72:eaff:fead	F8:72:EA:AD:C4:7C	eth1/1	lla-mac-match
stale			:c47c			,trusted-access
1/104	nd	ipv6	2001:0:0:200::10	F8:72:EA:AD:C4:7C	eth1/1	lla-mac-match
stale						,trusted-access
1/104	local	ipv6	fe80::200	00:22:BD:F8:19:FF	vlan4	static-
reach						authenticated
able						authenticated
1/104	local	ipv6	2001:0:0:200::1	00:22:BD:F8:19:FF	vlan4	static-
reach						authenticated
able						authenticated
Pod/Node	Type	IP Address	Creation TS	Last Refresh TS		
Lease Period						
-----	-----	-----	-----	-----		
1/102	local	192.0.200.1	2017-07-20T04:22:38.000+00:00	2017-07-20T04:22:38.000+00:00		
1/102	local	fe80::200	2017-07-20T04:22:56.000+00:00	2017-07-20T04:22:56.000+00:00		
1/102	local	2001:0:0:200::1	2017-07-20T04:22:57.000+00:00	2017-07-20T04:22:57.000+00:00		
1/101	arp	192.0.200.23	2017-07-27T10:55:20.000+00:00	2017-07-27T16:07:24.000+00:00		
1/101	local	192.0.200.1	2017-07-27T10:48:09.000+00:00	2017-07-27T10:48:09.000+00:00		
1/101	nd	fe80::d272:dcff:fea0	2017-07-27T10:52:16.000+00:00	2017-07-27T16:04:29.000+00:00		
		:261				
1/101	nd	2001:0:0:200::20	2017-07-27T10:57:32.000+00:00	2017-07-27T16:07:24.000+00:00		
1/101	nd	2001::200:d272:dcff:	2017-07-27T11:21:45.000+00:00	2017-07-27T16:07:24.000+00:00		
		fea0:261				
1/101	local	fe80::200	2017-07-27T10:48:10.000+00:00	2017-07-27T10:48:10.000+00:00		
1/101	local	2001:0:0:200::1	2017-07-27T10:48:11.000+00:00	2017-07-27T10:48:11.000+00:00		
1/103	local	192.0.200.1	2017-07-26T22:03:56.000+00:00	2017-07-26T22:03:56.000+00:00		
1/103	local	fe80::200	2017-07-26T22:03:57.000+00:00	2017-07-26T22:03:57.000+00:00		
1/103	local	2001:0:0:200::1	2017-07-26T22:03:58.000+00:00	2017-07-26T22:03:58.000+00:00		
1/104	arp	192.0.200.10	2017-07-27T11:21:13.000+00:00	2017-07-27T16:05:48.000+00:00		
1/104	arp	172.29.207.222	2017-07-27T11:54:48.000+00:00	2017-07-27T16:06:38.000+00:00		
1/104	local	192.0.200.1	2017-07-27T10:49:13.000+00:00	2017-07-27T10:49:13.000+00:00		
1/104	nd	fe80::fa72:eaff:fead	2017-07-27T11:21:13.000+00:00	2017-07-27T16:06:43.000+00:00		
		:c47c				
1/104	nd	2001:0:0:200::10	2017-07-27T11:21:13.000+00:00	2017-07-27T16:06:19.000+00:00		

```

1/104      local    fe80::200          2017-07-27T10:49:14.000+00:00  2017-07-27T10:49:14.000+00:00
1/104      local    2001:0:0:200::1      2017-07-27T10:49:15.000+00:00  2017-07-27T10:49:15.000+00:00

```

```
swtb23-ifc1#
```

```
swtb23-ifc1# show tenant t0 bridge-domain bd200 first-hop-security statistics arp
```

```

Pod/Node      : 1/101
Request Received : 4
Request Switched : 2
Request Dropped : 2
Reply Received  : 257
Reply Switched  : 257
Reply Dropped   : 0

```

```

Pod/Node      : 1/104
Request Received : 6
Request Switched : 6
Request Dropped : 0
Reply Received  : 954
Reply Switched  : 954
Reply Dropped   : 0

```

```
swtb23-ifc1# show tenant t0 bridge-domain bd200 first-hop-security statistics dhcpv4
```

```

Pod/Node      : 1/102
Discovery Received : 5
Discovery Switched : 5
Discovery Dropped : 0
Offer Received    : 0
Offer Switched    : 0
Offer Dropped     : 0
Request Received   : 0
Request Switched   : 0
Request Dropped    : 0
Ack Received       : 0
Ack Switched       : 0
Ack Dropped        : 0
Nack Received      : 0
Nack Switched      : 0
Nack Dropped       : 0
Decline Received   : 0
Decline Switched   : 0
Decline Dropped    : 0
Release Received   : 0
Release Switched   : 0
Release Dropped    : 0
Information Received : 0
Information Switched : 0
Information Dropped : 0
Lease Query Received : 0
Lease Query Switched : 0
Lease Query Dropped : 0
Lease Active Received : 0
Lease Active Switched : 0
Lease Active Dropped : 0
Lease Unassignment Received : 0
Lease Unassignment Switched : 0
Lease Unassignment Dropped : 0
Lease Unknown Received : 0
Lease Unknown Switched : 0
Lease Unknown Dropped : 0

```

```
swtb23-ifc1# show tenant t0 bridge-domain bd200 first-hop-security statistics neighbor-discovery
```

```

Pod/Node : 1/101
Neighbor Solicitation Received : 125
Neighbor Solicitation Switched : 121
Neighbor Solicitation Dropped : 4
Neighbor Advertisement Received : 519
Neighbor Advertisement Switched : 519
Neighbor Advertisement Drop : 0
Router Solicitation Received : 4
Router Solicitation Switched : 4
Router Solicitation Dropped : 0
Router Adv Received : 0
Router Adv Switched : 0
Router Adv Dropped : 0
Redirect Received : 0
Redirect Switched : 0
Redirect Dropped : 0

```

```

Pod/Node : 1/104
Neighbor Solicitation Received : 123
Neighbor Solicitation Switched : 47
Neighbor Solicitation Dropped : 76
Neighbor Advertisement Received : 252
Neighbor Advertisement Switched : 228
Neighbor Advertisement Drop : 24
Router Solicitation Received : 0
Router Solicitation Switched : 0
Router Solicitation Dropped : 0
Router Adv Received : 53
Router Adv Switched : 6
Router Adv Dropped : 47
Redirect Received : 0
Redirect Switched : 0
Redirect Dropped : 0

```

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 NX-OS Style CLI

Step 1 Configure the radius authentication group:

Example:

```

apic1# configure
apic1(config)#
apic1(config)# aaa group server radius myradiusgrp
apic1(config-radius)#server 192.168.0.100 priority 1
apic1(config-radius)#exit

```

Step 2 Configure node level port authentication policy:

Example:

```

apic1(config)# policy-map type port-authentication mydot1x
apic1(config-pmap-port-authentication)#radius-provider-group myradiusgrp
apic1(config-pmap-port-authentication)#fail-auth-vlan 2001
apic1(config-pmap-port-authentication)#fail-auth-epg tenant tn1 application ap1 epg epg256
apic1(config)# exit

```

Step 3 Configure policy group and specify port authentication policy in the group:

Example:

```

apic1(config)#template leaf-policy-group lpg2
apic1(config-leaf-policy-group)# port-authentication mydot1x
apic1(config-leaf-policy-group)#exit

```

Step 4 Configure the leaf switch profile:

Example:

```

apic1(config)# leaf-profile mylp2
apic1(config-leaf-profile)#leaf-group mylg2
apic1(config-leaf-group)# leaf-policy-group lpg2
apic1(config-leaf-group)#exit

```

Configuring 802.1X Port Authentication Using the NX-OS Style CLI

Step 1 Configure a Policy Group:

Example:

```

apic1# configure
apic1(config)#
apic1(config)# template policy-group mypol
apic1(config-pol-grp-if)# switchport port-authentication mydot1x
apic1(config-port-authentication)# host-mode multi-host
apic1(config-port-authentication)# no shutdown
apic1(config-port-authentication)# exit
apic1(config-pol-grp-if)# exit

```

Step 2 Configure the leaf interface profile:

Example:

```

apic1(config)#
apic1(config)#leaf-interface-profile myprofile
apic1(config-leaf-if-profile)#leaf-interface-group mygroup
apic1(config-leaf-if-group)# interface ethernet 1/10-12
apic1(config-leaf-if-group)# policy-group mypol
apic1(config-leaf-if-group)# exit
apic1(config-leaf-if-profile)# exit

```


Step 3 Configure the leaf profile:**Example:**

```
apic1(config)#  
apic1(config)# leaf-profile myleafprofile  
apic1(config-leaf-profile)# leaf-group myleafgrp  
apic1(config-leaf-group)# leaf 101  
apic1(config-leaf-group)# exit
```

Step 4 Apply an interface policy on the leaf switch profile:**Example:**

```
apic1(config-leaf-profile)# leaf-interface-profile myprofile  
apic1(config-leaf-group)# exit
```



CHAPTER 11

Configuring Anycast Services

This chapter contains the following sections:

- [About Anycast Services, on page 423](#)
- [Configuring Anycast Services Using the NX-OS Style CLI, on page 423](#)

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 NX-OS Style CLI

These examples show how to configure Anycast services in three methods, using the NX-OS style CLI:

- 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 Configure Anycast services behind an EPG subnet, using the following commands:

a) **configure**

Enters configuration mode.

Example:

```
apic1# configure
```

b) **tenant** *tenant-name*

Creates a tenant if it does not exist or enters tenant configuration mode.

Example:

```
apic1(config)# tenant anycast1-it
```

c) **application** *app-name*

Creates an application profile if it doesn't exist and enters application profile configuration mode.

Example:

```
apic1(config-tenant)# application AP0
```

d) **epg** *epg-name*

Creates an EPG if it doesn't exist and enters EPG configuration mode.

Example:

```
apic1(config-tenant-app)# epg epg1
```

e) **endpointip** *ip-address***anycast** *mac-address*

Configures the Anycast IP address with netmask and MAC address for the Anycast service behind the EPG. The Anycast subnet must have a /32 or /128 netmask.

Example:

```
apic1(config-tenant-app-epg)# endpoint ip 1.2.3.4/32 anycast 00:11:22:33:44:55
```

Step 2 Configure Anycast for Layer 4 to Layer 7 services with PBR, using the following commands:

a) **configure**

Enters configuration mode.

Example:

```
apic1# configure
```

b) **tenant** *name*

Creates a tenant if it does not exist or enters tenant configuration mode.

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

c) **svcredir-pol** *name*

Enters service-redirect policy mode and creates a service redirection policy.

Example:

```
apic1(config-tenant)# svcredir-pol N1Ext
```

d) **anycast enable**

Enables Anycast for the service redirection policy. Use the no form of the command to disable Anycast for the policy.

Example:

```
apic1(svcredir-pol)# anycast enable
```

e) **redir-dest** *ip-addr mac-addr*

Defines the Anycast IP and MAC addresses for the Layer 4 to Layer 7 service redirection policy.

Example:

```
apic1(svcredir-pol)# redir-dest 2000::25 00:00:00:00:00:07
```

Step 3

Configure Anycast for Layer 4 to Layer 7 services without PBR, with the following commands:

a) **configure**

Enters configuration mode.

Example:

```
apic1# configure
```

b) **tenant** *name*

Creates a tenant if it does not exist or enters tenant configuration mode.

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

c) **l4l7 graph** *connector-name contract name*

Creates a Layer 4 to Layer 7 service graph associated with a contract.

Example:

```
apic1(config-tenant)# l4l7 graph WebGraph contract default
```

d) **service** *device-cluster-name*

Defines the service with a device cluster.

Example:

```
apic1(config-graph)# service N1
```

e) **connector** *name* [**cluster-interface** *cluster-interface-name*]

Enters connector configuration mode and defines the device cluster interface.

Example:

```
apic1(config-service)# connector provider
```

f) **subnet-ip** *IP-addr_with_netmask* **subnet-ctrl no-default-gateway**

Defines the Anycast IP address (with /32 netmask and the subnet control, no-default-gateway). To remove it, use the no form of the command.

Example:

```
apicl(config-connector)# subnet-ip 50.50.50.50/32 subnet-ctrl no-default-gateway
```

g) **mac-address** *mac-address*

Defines the Anycast MAC address. To remove it, use the no form of the command.

Example:

```
apicl(config-subnet-ip)# mac-address 00.00.00.00.00.50
```

Example

The following example configures Anycast services behind EPG1:

```
apicl# configure
apicl(config)# tenant anycast1-it
apicl(config-tenant)# application AP0
apicl(config-tenant-app)# epg epg-1
apicl(config-tenant-app-epg)# endpoint ip 1.2.3.4/32 anycast 00:11:22:33:44:55
```

The following example configures Anycast services in a Layer 4 to Layer 7 service redirection policy:

```
apicl# configure
apicl(config)# tenant t1
apicl(config-tenant)# svcdir-pol N1Ext
apicl(svcdir-pol)# anycast enable
apicl(svcdir-pol)# redir-dest 2000::25 00:00:00:00:00:07
```

The following example configures Anycast services in a Layer 4 to Layer 7 service without PBR:

```
apicl# configure
apicl(config)# tenant t1
apicl(config-tenant)# l4l7 graph WebGraph contract default
apicl(config-graph)# service N1
apicl(config-service)# connector provider
apicl(config-connector)# subnet-ip 50.50.50.50/32 subnet-ctrl no-default-gateway
apicl(config-subnet-ip)# mac-address 00.00.00.00.00.50
```



CHAPTER 12

Configuring VMM

- [Configuring VMM, on page 427](#)

Configuring VMM

For information about configuring virtual machine management using the NX-OS style CLI for APIC, see the *Cisco ACI Virtualization Guide*.



CHAPTER 13

Configuring Layer 4 to Layer 7 Services

- [Configuring Layer 4 to Layer 7 Services, on page 429](#)

Configuring Layer 4 to Layer 7 Services

For information about configuring Layer 4 to Layer 7 services using the NX-OS-style CLI for Cisco Application Policy Infrastructure Controller (Cisco APIC), see the *Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*.



CHAPTER 14

Configuring Global Policies

- [About Global Policies, on page 431](#)
- [Configuring Out-of-Band Management NTP, on page 431](#)
- [Configuring the System Clock, on page 434](#)
- [Configuring Error Disable Recovery, on page 435](#)
- [Configuring Link Level Discovery Protocol, on page 436](#)
- [Configuring Miscabling Protocol, on page 437](#)
- [Configuring the Endpoint Loop Protection Policy, on page 439](#)
- [Configuring the Rogue Endpoint Control Policy, on page 440](#)
- [Configuring IP Aging, on page 441](#)
- [Configuring the Dynamic Load Balancer, on page 442](#)
- [Configuring Spanning Tree Protocol, on page 443](#)
- [Configuring IS-IS, on page 445](#)
- [Configuring BGP Route Reflectors, on page 447](#)
- [Decommissioning a Node, on page 448](#)
- [Configuring Power Management, on page 449](#)
- [Configuring a Scheduler, on page 451](#)
- [Configuring System MTU, on page 453](#)
- [About PTP, on page 454](#)

About Global Policies

The APIC fabric has many fabric level configurations, which are applied to the entire fabric components (switches and ports). In some cases, lower level policies (switch or interface level) exist to override these policies. For example, while MCP policy can enable the MCP feature globally, an interface level MCP policy exists to enable or disable MCP on an individual interface.

Configuring Out-of-Band Management NTP

When an ACI fabric is deployed with out-of-band management, each node of the fabric is managed from outside the ACI fabric. You can configure an out-of-band management NTP server so that each node can individually query the same NTP server as a consistent clock source.

SUMMARY STEPS

1. **configure**
2. **template ntp-fabric** *ntp-fabric-template-name*
3. **[no] server** *dns-name-or-ipaddress* **[prefer]** **[use-vrf {inband-mgmt | oob-default}]** **[key key-value]**
4. **[no] authenticate**
5. **[no] authentication-key** *key-value*
6. **[no] trusted-key** *key-value*
7. **exit**
8. **template pod-group** *pod-group-template-name*
9. **inherit ntp-fabric** *ntp-fabric-template-name*
10. **exit**
11. **pod-profile** *pod-profile-name*
12. **Pods** {*pod-range-1-255* | **all**}
13. **inherit pod-group** *pod-group-name*
14. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters configuration mode.
Step 2	template ntp-fabric <i>ntp-fabric-template-name</i> Example: apic1(config)# template ntp-fabric poll	Specifies the NTP template (policy) for the fabric.
Step 3	[no] server <i>dns-name-or-ipaddress</i> [prefer] [use-vrf {inband-mgmt oob-default}] [key key-value] Example: apic1(config-template-ntp-fabric)# server 192.0.20.123 prefer use-vrf oob-mgmt	Configures an NTP server for the active NTP policy. To make this server the preferred server for the active NTP policy, include the prefer keyword. If NTP authentication is enabled, specify a reference key ID. To specify the in-band or out-of-band management access VRF, include the use-vrf keyword with the inb-default or oob-default keyword.
Step 4	[no] authenticate Example: apic1(config-template-ntp-fabric)# no authenticate	Enables (or disables) NTP authentication.
Step 5	[no] authentication-key <i>key-value</i> Example: apic1(config-template-ntp-fabric)# authentication-key 12345	Configures an authentication NTP authentication. The range is 1 to 65535.
Step 6	[no] trusted-key <i>key-value</i> Example:	Configures a trusted NTP authentication. The range is 1 to 65535.

	Command or Action	Purpose
	<code>apicl (config-template-ntp-fabric) # trusted-key 54321</code>	
Step 7	exit Example: <code>apicl (config-template-ntp-fabric) # exit</code>	Returns to global configuration mode
Step 8	template pod-group <i>pod-group-template-name</i> Example: <code>apicl (config) # template pod-group allPods</code>	Configures a pod-group template (policy).
Step 9	inherit ntp-fabric <i>ntp-fabric-template-name</i> Example: <code>apicl (config-pod-group) # inherit ntp-fabric poll</code>	Configures the NTP fabric pod-group to use the previously configured NTP fabric template (policy).
Step 10	exit Example: <code>apicl (config-template-pod-group) # exit</code>	Returns to global configuration mode
Step 11	pod-profile <i>pod-profile-name</i> Example: <code>apicl (config) # pod-profile all</code>	Configures a pod profile.
Step 12	pods {<i>pod-range-1-255</i> all} Example: <code>apicl (config-pod-profile) # pods all</code>	Configures a set of pods.
Step 13	inherit pod-group <i>pod-group-name</i> Example: <code>apicl (config-pod-profile-pods) # inherit pod-group allPods</code>	Associates the pod-profile with the previously configured pod group.
Step 14	end Example: <code>apicl (config-pod-profile-pods) # end</code>	Returns to EXEC mode.

Examples

This example shows how to configure a preferred out-of-band NTP server and how to verify the configuration and deployment.

```

apicl# configure t
apicl (config) # template ntp-fabric poll
apicl (config-template-ntp-fabric) # server 192.0.20.123 use-vrf oob-default
apicl (config-template-ntp-fabric) # no authenticate
apicl (config-template-ntp-fabric) # authentication-key 12345
apicl (config-template-ntp-fabric) # trusted-key 12345

```

```

apicl(config-template-ntp-fabric)# exit
apicl(config)# template pod-group allPods
apicl(config-pod-group)# inherit ntp-fabric poll
apicl(config-pod-group)# exit
apicl(config)# pod-profile all
apicl(config-pod-profile)# pods all
apicl(config-pod-profile-pods)# inherit pod-group allPods
apicl(config-pod-profile-pods)# end
apicl#

```

```

apicl# show ntpq
nodeid      remote          refid    st    t    when    poll    reach    delay    offset    jitter
-----  -  -----
1          *  192.0.20.123  .GPS.    u    27     64     377     76.427    0.087    0.067
2          *  192.0.20.123  .GPS.    u     3     64     377     75.932    0.001    0.021
3          *  192.0.20.123  .GPS.    u     3     64     377     75.932    0.001    0.021

```

Configuring the System Clock

SUMMARY STEPS

1. configure
2. [no] clock display-format {local | utc}
3. [no] clock show-offset enable
4. [no] clock timezone *timezone-code*
5. show clock

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	[no] clock display-format {local utc} Example: apicl(config)# clock display-format local	Sets the clock date time format to either local or UTC time.
Step 3	[no] clock show-offset enable Example: apicl(config)# clock show-offset enable	Enables (or disables) the display of the offset from UTC. This setting is valid only when the display-format is local.
Step 4	[no] clock timezone <i>timezone-code</i> Example:	Specifies the local time zone. The default is p0_utc.

	Command or Action	Purpose
	<code>apic1(config)# clock timezone n420_America-Los_Angeles</code>	
Step 5	show clock Example: <code>apic1(config)# show clock</code>	Specifies the delay time for LLDP to initialize on any interface . The range is 1 to 10 seconds; the default is 2 seconds.

Examples

This example shows how to configure the system clock for local time in the Los Angeles timezone.

```
apic1# configure terminal
apic1(config)# clock display-format local
apic1(config)# clock show-offset enable
apic1(config)# clock timezone n420_America-Los_Angeles
apic1(config)# show clock
Time : 20:47:37.038 UTC-08:00 Sun Nov 08 2015
```

Configuring Error Disable Recovery

The error disabled recovery (EDR) policy is a fabric level policy that can enable ports that loop detection and BPDU policies disabled after an interval that the administrator can configure.

SUMMARY STEPS

1. **configure**
2. **[no] errdisable recovery interval** *seconds*
3. **[no] errdisable recovery cause** {*bpduguard* | *ep-move* | *mcp-loop*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] errdisable recovery interval <i>seconds</i> Example: <code>apic1(config)# errdisable recovery interval 300</code>	Specifies the interval for an interface to recover from the error-disabled state. The range is from 30 to 65535 seconds
Step 3	[no] errdisable recovery cause { <i>bpduguard</i> <i>ep-move</i> <i>mcp-loop</i> } Example: <code>apic1(config)# errdisable recovery cause mcp-loop</code>	Specifies a condition under which the interface automatically recovers from the error-disabled state, and the device retries bringing the interface up. The default is disabled. The condition options are:

	Command or Action	Purpose
		• bpduguard—Enable timer to recover from a BPDU guard error disable. • ep-move—Enable timer to recover from an endpoint move error disable. • mcp-loop—Enable timer to recover from an MCP loop error disable. • storm-control-recovery—Enable timer to recover from a storm control recovery error disable.

Examples

This example shows how to configure EDR to recover from an MCP loop error disable.

```
apicl# configure terminal
apicl(config)# errdisable recovery interval 300
apicl(config)# errdisable recovery cause mcp-loop
```

Configuring Link Level Discovery Protocol

The Link Layer Discovery Protocol (LLDP) is a device discovery protocol that allows network devices to advertise information about themselves to other devices on the network. LLDP determines the layer 2 connectivity between switches.

SUMMARY STEPS

1. **configure**
2. **[no] lldp holdtime** *seconds*
3. **[no] lldp holdtime** *seconds*
4. **[no] lldp reinit** *seconds*
5. **[no] lldp timer** *seconds*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	[no] lldp holdtime <i>seconds</i> Example: apicl(config)# lldp holdtime	Specifies the hold time to be sent in LLDP packets.

	Command or Action	Purpose
Step 3	[no] lldp holdtime <i>seconds</i> Example: <code>apic1(config)# lldp holdtime 120</code>	Specifies the hold time to be sent in LLDP packets. The range is 10 to 255 seconds; the default is 120 seconds.
Step 4	[no] lldp reinit <i>seconds</i> Example: <code>apic1(config)# lldp reinit 2</code>	Specifies the delay time for LLDP to initialize on any interface . The range is 1 to 10 seconds; the default is 2 seconds.
Step 5	[no] lldp timer <i>seconds</i> Example: <code>apic1(config)# lldp timer 30</code>	Specifies the transmission frequency seconds of LLDP updates in seconds. The range is 5 to 254 seconds; the default is 30 seconds.

Examples

This example shows how to configure LLDP.

```
apic1# configure terminal
apic1(config)# lldp holdtime 120
apic1(config)# lldp reinit 2
apic1(config)# lldp timer 30
```

Configuring Miscabling Protocol

The ACI fabric provides loop detection policies that can detect loops in Layer 2 network segments that are connected to ACI access ports. The ACI fabric implements the mis-cabling protocol (MCP), a fabric level policy that allows provisioning of MCP parameters as well as determining the port behavior if mis-cabling is detected. MCP works in a complementary manner with STP that is running on external Layer 2 networks, and handles Bridge Protocol Data Unit (BPDU) packets that access ports receive.

A fabric administrator provides a key that MCP uses to identify which MCP packets are initiated by the ACI fabric. The administrator can choose how the MCP policies identify loops and how to act upon the loops: syslog only, or disable the port.

SUMMARY STEPS

1. **configure**
2. **[no] mcp action port-disable**
3. **[no] mcp enable [key *key-value*]**
4. **[no] mcp factor *number***
5. **[no] mcp init-delay *seconds***
6. **[no] mcp transmit-frequency *frequency***

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	[no] mcp action port-disable Example: apicl(config)# mcp action port-disable	Specifies whether a port should be place in a disabled state if mis-cabling is detected.
Step 3	[no] mcp enable [key key-value] Example: apicl(config)# mcp enable key 0123456789abcdef	Allows enabling or disabling of the MCP protocol globally for the entire fabric. A password (key) is required when enabling the policy but not when disabling.
Step 4	[no] mcp factor <i>number</i> Example: apicl(config)# mcp factor 64	Sets the loop detection multiplication factor, which is used while sending MCP packets. The range is 1 to 255.
Step 5	[no] mcp init-delay <i>seconds</i> Example: apicl(config)# mcp init-delay 180	Specifies the initial delay time. The range is 0 to 1800 seconds; the default is 180.
Step 6	[no] mcp transmit-frequency <i>frequency</i> Example: apicl(config)# mcp transmit-frequency 2	Sets the frequency of transmission of MCP packets to detect mis-cabling. The range is 100 milliseconds to 300 seconds; the default is 2 seconds.

Examples

This example shows how to configure MCP for a transmit frequency of 2 seconds.

```

apicl# configure terminal
apicl(config)# mcp action port-disable
apicl(config)# mcp enable key 0123456789abcdef
apicl(config)# mcp factor 64
apicl(config)# mcp init-delay 180
apicl(config)# mcp transmit-frequency 2

```

This example shows how to configure MCP for a transmit frequency of 2 seconds and 300 milliseconds.

```

apicl# configure terminal
apicl(config)# mcp action port-disable
apicl(config)# mcp enable key 0123456789abcdef
apicl(config)# mcp factor 64
apicl(config)# mcp init-delay 180
apicl(config)# mcp transmit-frequency 2 300

```

Configuring the Endpoint Loop Protection Policy

The endpoint loop protection policy is a fabric level policy used in detection of frequent endpoint (host) moves from one fabric port to another. The policy configures what action is to be taken if such an event is detected.

SUMMARY STEPS

1. **configure**
2. **[no] endpoint loop-detect action {bd-learn-disable | port-disable}**
3. **[no] endpoint loop-detect enable**
4. **[no] endpoint loop-detect factor *number***
5. **[no] endpoint loop-detect interval *seconds***

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] endpoint loop-detect action {bd-learn-disable port-disable} Example: <code>apic1(config)# endpoint loop-detect action port-disable</code>	Specifies the action to perform when an endpoint loop is detected. The options are: • bd-learn-disable—Disable MAC address learning on the bridge domain. • port-disable—Disable the port.
Step 3	[no] endpoint loop-detect enable Example: <code>apic1(config)# endpoint loop-detect enable</code>	Allows enabling or disabling of the endpoint loop protection protocol globally for the entire fabric.
Step 4	[no] endpoint loop-detect factor <i>number</i> Example: <code>apic1(config)# endpoint loop-detect factor 64</code>	Sets the loop detection multiplication factor. The range is 1 to 255.
Step 5	[no] endpoint loop-detect interval <i>seconds</i> Example: <code>apic1(config)# endpoint loop-detect interval 60</code>	Specifies the loop detection interval. The range is 30 to 300 seconds.

Examples

This example shows how to configure the endpoint loop protection policy.

```
apic1# configure terminal
apic1(config)# endpoint loop-detect action port-disable
```

```
apicl(config)# endpoint loop-detect enable
apicl(config)# endpoint loop-detect factor 64
apicl(config)# endpoint loop-detect interval 60
```

Configuring the Rogue Endpoint Control Policy

About the Rogue Endpoint Control Policy

A rogue endpoint attacks top of rack (ToR) switches through frequently, repeatedly injecting packets on different ToR 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)
- 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
- 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 Rogue Endpoint Control Using the NX-OS Style CLI

You can configure the **Rogue EP Control** policy for the fabric, to detect and delete unauthorized endpoints, using the NX-OS style CLI.

Step 1 **configure**

Enters global configuration mode.

Example:

```
apicl# configure
```

Step 2 **endpoint rogue-detect enable**

Enables the global Rogue Endpoint Control policy.

Example:

```
apic1(config)# endpoint rogue-detect enable
```

Step 3 **endpoint rogue-detect hold-interval** *hold_interval*

Sets the hold interval in seconds after the endpoint is declared rogue, where it is kept static so learning is prevented, and the traffic to and from the rogue endpoint is dropped. After this interval, the endpoint is deleted. Valid values are from 1800 to 3600 seconds. The default is 1800.

Example:

```
apic1(config)# endpoint rogue-detect hold-interval 1800
```

Step 4 **endpoint rogue-detect interval** *interval*

Sets the rogue detection interval in seconds, which specifies the time to detect rogue endpoints. Valid values are from 0 to 65535 seconds. The default is 60.

Example:

```
apic1(config)# endpoint rogue-detect interval 60
```

Step 5 **endpoint rogue-detect factor** *factor*

Specifies the multiplication factor for determining if an endpoint is unauthorized. If the endpoint moves more times during the interval, the EP is declared rogue. Valid values are from 2 to 10. The default is 6.

Example:

```
apic1# endpoint rogue-detect factor 6
```

Step 6 This example configures a Rogue Endpoint Control policy.**Example:**

```
apic1# cconfigure
apic1(config)# endpoint rogue-detect enable
apic1(config)# endpoint rogue-detect hold-interval 1800
apic1(config)# endpoint rogue-detect interval 60
apic1(config)# endpoint rogue-detect factor 6
```

Configuring IP Aging

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 the IP Aging Policy Using the NX-OS-Style CLI

This section explains how to enable and disable the IP Aging policy using the CLI.

Step 1 To enable the IP aging policy:

Example:

```
ifc1(config)# endpoint ip aging
```

Step 2 To disable the IP aging policy:

Example:

```
ifav9-ifc1(config)# no endpoint ip aging
```

What to do next

To specify the interval used for tracking IP addresses on endpoints, create an Endpoint Retention policy.

Configuring the Dynamic Load Balancer

Dynamic load balancing (DLB) adjusts the traffic allocations according to congestion levels. DLB measures the congestion across the available paths and places the flows on the least congested paths, which results in an optimal or near optimal placement of the data.

DLB can be configured to place traffic on the available uplinks using the granularity of flows or flowlets. Flowlets are bursts of packets from a flow that are separated by suitably large gaps in time. If the idle interval between two bursts of packets is larger than the maximum difference in latency among available paths, the second burst (or flowlet) can be sent along a different path than the first without reordering packets. This idle interval is measured with a timer called the flowlet timer. Flowlets provide a higher granular alternative to flows for load balancing without causing packet reordering.

SUMMARY STEPS

1. **configure**
2. **[no] system dynamic-load-balance mode {dynamic-aggressive | dynamic-conservative | link-failure-resiliency | packet-prioritization}**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	[no] system dynamic-load-balance mode {dynamic-aggressive dynamic-conservative link-failure-resiliency packet-prioritization}	Specifies the mode of operation of the load balancer. The modes are: • dynamic-aggressive—The flowlet timeout is a relatively small value. This very fine-grained dynamic load balancing is optimal for the distribution of traffic, but some packet reordering might occur. However, the

	Command or Action	Purpose
		overall benefit to application performance is equal to or better than the conservative mode. • dynamic-conservative—The flowlet timeout is a larger value that guarantees packets are not to be re-ordered. The tradeoff is less granular load balancing because new flowlet opportunities are less frequent. • link-failure-resiliency—Static load balancing gives a distribution of flows across the available links that is roughly even. • packet-prioritization—Dynamic Packet Prioritization (DPP) prioritizes short flows higher than long flows; a short flow is less than approximately 15 packets. Because short flows are more sensitive to latency than long ones, DPP can improve overall application performance. <pre>apic1(config)# system dynamic-load-balance mode packet-prioritization</pre>

Examples

This example shows how to configure dynamic load balancing with packet prioritization.

```
apic1# configure terminal
apic1(config)# system dynamic-load-balance mode packet-prioritization
```

Configuring Spanning Tree Protocol

Multiple spanning-tree (MST) enables multiple VLANs to be mapped to the same spanning-tree instance, reducing the number of spanning-tree instances needed to support a large number of VLANs.



Note Multiple Spanning Tree (MST) is not supported on interfaces configured with the Per Port VLAN feature (configuring multiple EPGs on a leaf switch using the same VLAN ID with localPort scope).

SUMMARY STEPS

1. **configure**
2. **spanning-tree mst configuration**
3. **[no] bpdu-filter**
4. **[no] region *region-name***
5. **[no] instance *instance-id* vlan *vlan-range***

6. *revision number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	spanning-tree mst configuration Example: apicl(config)# spanning-tree mst configuration	Enters global configuration mode.
Step 3	[no] bpdu-filter Example: apicl(config-stp)# bpdu-filter	Enters global configuration mode.
Step 4	[no] region <i>region-name</i> Example: apicl(config-stp)# region region1	For switches to participate in multiple spanning-tree (MST) instances, you must consistently configure the switches with the same MST configuration information. A collection of interconnected switches that have the same MST configuration comprises an MST region. Each region can support up to 65 spanning-tree instances.
Step 5	[no] instance <i>instance-id</i> vlan <i>vlan-range</i> Example: apicl(config-stp-region)# instance 2 vlan 1-63	Maps VLANs to an MST instance. You can assign a VLAN to only one spanning-tree instance at a time. The instance ID range is 1 to 4094. To specify a VLAN range, use a hyphen.
Step 6	revision <i>number</i> Example: apicl(config-stp-region)# revision 16	Specifies the configuration revision number. The range is 0 to 65535.

Examples

This example shows how to configure an MST spanning-tree policy.

```

apicl# configure terminal
apicl(config)# spanning-tree mst configuration
apicl(config-stp)# bpdu-filter
apicl(config-stp)# region region1
apicl(config-stp-region)# instance 2 vlan 1-63
apicl(config-stp-region)# revision 16

```


Configuring IS-IS

Intermediate System-to-Intermediate System (IS-IS) is a dynamic link-state routing protocol that can detect changes in the network topology and calculate loop-free routes to other nodes in the network.

SUMMARY STEPS

1. **configure**
2. **template isis-fabric** *isis-fabric-template-name*
3. **[no] lsp-fast-flood**
4. **[no] lsp-gen-interval level-1** *lsp-max-wait* [*lsp-initial-wait* *lsp-second-wait*]
5. **[no] lsp-mtu** *mtu*
6. **[no] spf-interval level-1** *spf-max-wait* [*spf-initial-wait* *spf-second-wait*]
7. **exit**
8. **template pod-group** *pod-group-template-name*
9. **inherit pod-group** *pod-group-name*
10. **exit**
11. **pod-profile** *pod-profile-name*
12. **pods** {*pod-range-1-255* | **all**}
13. **inherit pod-group** *pod-group-name*
14. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	template isis-fabric <i>isis-fabric-template-name</i> Example: <code>apic1(config)# template isis-fabric polIsIs</code>	Enters Intermediate System-to-Intermediate System (IS-IS) configuration mode and creates an IS-IS fabric template (policy).
Step 3	[no] lsp-fast-flood Example: <code>apic1(config-template-isis-fabric)# lsp-fast-flood</code>	Enables the fast-flood feature, which improves convergence time when new link-state packets (LSPs) are generated in the network and shortest path first (SPF) is triggered by the new LSPs. We recommend that you enable the fast-flooding of LSPs before the router runs the SPF computation, to ensure that the whole network achieves a faster convergence time.
Step 4	[no] lsp-gen-interval level-1 <i>lsp-max-wait</i> [<i>lsp-initial-wait</i> <i>lsp-second-wait</i>] Example: <code>apic1(config-template-isis-fabric)# lsp-gen-interval level-1 500 500 500</code>	Configures the IS-IS throttle for LSP generation. The parameters are as follows: <i>lsp-max-wait</i>—The maximum wait between the trigger and LSP generation.

	Command or Action	Purpose
		• <i>lsp-initial-wait</i>—The initial wait between the trigger and LSP generation. • <i>lsp-second-wait</i>—The second wait used for LSP throttle during backoff. The <i>lsp-max-wait</i> parameter is required. The other two parameters are optional but must appear together. The range for each is 50 to 120000 milliseconds.
Step 5	[no] lsp-mtu <i>mtu</i> Example: <pre>apic1(config-template-isis-fabric)# lsp-mtu 2048</pre>	Sets the maximum transmission unit (MTU) size of IS-IS hello packets. The range is 256 to 4352. IS-IS hello packets are used to discover and maintain adjacencies. By default, the hello packets are padded to the full maximum transmission unit (MTU) size to allow for early detection of errors due to transmission problems with large frames or due to mismatched MTUs on adjacent interfaces. However, IS-IS adjacency formation may fail due to MTU mismatch on a link, requiring the adjustment of the MTU size.
Step 6	[no] spf-interval level-1 <i>spf-max-wait</i> [<i>spf-initial-wait</i> <i>spf-second-wait</i>] Example: <pre>apic1(config-template-isis-fabric)# spf-interval level-1 500 500 500</pre>	Configures the interval between LSA arrivals. The parameters are as follows: • <i>spf-max-wait</i>—The maximum wait between the trigger and SPF computation. • <i>spf-initial-wait</i>—The initial wait between the trigger and SPF computation. • <i>spf-second-wait</i>—The second wait used for SPF computation during backoff. The <i>spf-max-wait</i> parameter is required. The other two parameters are optional but must appear together. The range for each is 50 to 120000 milliseconds.
Step 7	exit Example: <pre>apic1(config-template-isis-fabric)# exit</pre>	Returns to global configuration mode.
Step 8	template pod-group <i>pod-group-template-name</i> Example: <pre>apic1(config)# template pod-group allPods</pre>	Creates a pod group template (policy).
Step 9	inherit pod-group <i>pod-group-name</i> Example: <pre>apic1(config-pod-group)# inherit isis-fabric polIsIs</pre>	Configures the template pod-group to use the previously configured isis-fabric template (policy).

	Command or Action	Purpose
Step 10	exit Example: <code>apic1(config-pod-group)# exit</code>	Returns to global configuration mode.
Step 11	pod-profile <i>pod-profile-name</i> Example: <code>apic1(config)# pod-profile all</code>	Configures a pod profile.
Step 12	pods { <i>pod-range-1-255</i> all } Example: <code>apic1(config-pod-profile)# pods all</code>	Configures a set of pods.
Step 13	inherit pod-group <i>pod-group-name</i> Example: <code>apic1(config-pod-profile-pods)# inherit pod-group allPods</code>	Configures the pod-profile to use the previously configured pod group.
Step 14	end Example: <code>apic1(config-pod-profile-pods)# end</code>	Returns to EXEC mode.

Examples

This example shows how to configure IS-IS.

```

aapic1# configure
apic1(config)# template isis-fabric polIsIs
apic1(config-template-isis-fabric)# lsp-fast-flood
apic1(config-template-isis-fabric)# lsp-gen-interval level-1 500 500 500
apic1(config-template-isis-fabric)# lsp-mtu 2048
apic1(config-template-isis-fabric)# spf-interval level-1 500 500 500
apic1(config-template-isis-fabric)# exit
apic1(config)# template pod-group allPods
apic1(config-pod-group)# inherit isis-fabric polIsIs
apic1(config-pod-group)# exit
apic1(config)# pod-profile all
apic1(config-pod-profile)# pods all
apic1(config-pod-profile-pods)# inherit pod-group allPods
apic1(config-pod-profile-pods)# end
apic1#

```

Configuring BGP Route Reflectors

The ACI fabric route reflectors use multiprotocol Border Gateway Protocol (MP-BGP) to distribute external routes within the fabric. To enable route reflectors in the ACI fabric, the fabric administrator must select the spine switches that will be the route reflectors, and provide the autonomous system (AS) number. For

redundancy purposes, more than one spine is configured as a router reflector node (one primary and one secondary reflector).

SUMMARY STEPS

1. **configure**
2. **bgp-fabric**
3. **asn *asn-value***
4. **[no] route-reflector spine *list***

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apicl# configure</code>	Enters global configuration mode.
Step 2	bgp-fabric Example: <code>apicl(config)# bgp-fabric</code>	Enters BGP configuration mode for the fabric.
Step 3	asn <i>asn-value</i> Example: <code>apicl(config-bgp-fabric)# asn 123456789</code>	Configures the BGP Autonomous System number (ASN), which uniquely identifies an autonomous system. The ASN is between 1 and 4294967295. We recommend that you enable the fast-flooding of LSPs before the router runs the SPF computation, to ensure that the whole network achieves a faster convergence time.
Step 4	[no] route-reflector spine <i>list</i> Example: <code>apicl(config-bgp-fabric)# route-reflector spine spine1,spine2</code>	Configure up to two spine nodes as route reflectors. For redundancy ,you should configure primary and secondary route reflectors.

Examples

This example shows how to configure spine1 and spine2 as BGP route reflectors.

```

apicl# configure
apicl(config)# bgp-fabric
apicl(config-bgp-fabric)# asn 123456789
apicl(config-bgp-fabric)# route-reflector spine spine1,spine2
apicl(config-bgp-fabric)# exit
apicl(config)#

```

Decommissioning a Node

Two levels of decommissioning are supported:

- Regular—Similar to disabling the node. After being decommissioned, the node cannot rejoin the fabric until the **no decommission** command is executed.
- Complete—When the node is decommissioned, all fabric configuration related to the node is cleared.

SUMMARY STEPS

1. **configure**
2. **[no] decommission {controller | switch} *node-id* [remove-from-controller]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] decommission {controller switch} <i>node-id</i> [remove-from-controller] Example: <code>apic1(config)# decommission switch 104 remove-from-controller</code>	Decommissions the specified node. Note that controller node ID numbers are between 1 and 100, while switch node ID numbers are between 101 and 4000.

Examples

This example shows how to perform a complete decommissioning of node 104 (a switch) and recommission node 5 (a controller), which was decommissioned with the regular level.

```
apic1# configure
apic1(config)# decommission switch 104 remove-from-controller
apic1(config)# no decommission controller 5
```

Configuring Power Management

SUMMARY STEPS

1. **configure**
2. **[no] power redundancy-policy *policy-name***
3. **[no] description *text***
4. **[no] redundancy-mode {combined | ps-redundant | redundant}**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apicl# configure</code>	Enters global configuration mode.
Step 2	[no] power redundancy-policy <i>policy-name</i> Example: <code>apicl(config)# power redundancy-policy myPowerPolicy</code>	Creates or configures a power supply redundancy policy.
Step 3	[no] description <i>text</i> Example: <code>apicl(config-power)# description 'This is my power redundancy policy'</code>	Adds a description for this power supply redundancy policy. If the text includes spaces, it must be enclosed in single quotes.
Step 4	[no] redundancy-mode {combined ps-redundant redundant} Example: <code>apicl(config-power)# redundancy-mode ps-redundant</code>	Specifies power supply redundancy mode. • combined— This mode does not provide power redundancy. The available power is the total power capacity of all power supplies. • ps-redundant—This mode provides an extra power supply in case an active power supply goes down. The power supply that can supply the most power operates in standby mode. The other one or two power supplies are active. The available power is the amount of power provided by the active power supply units. • redundant—This mode combines power supply redundancy and input source redundancy, which means that the chassis has an extra power supply and each half of each power supply is connected to one electrical grid while the other half of each power supply is connected to the other electrical grid. The available power is the lesser of the available power for power supply mode and input source mode.

Examples

This example shows how to configure a power supply redundancy policy for the ps-redundant mode.

```

apicl# configure
apicl(config)# power redundancy-policy myPowerPolicy
apicl(config-pod)# isis fabric
apicl(config-power)# description 'This is my power redundancy policy'
apicl(config-power)# redundancy-mode ps-redundant

```

Configuring a Scheduler

A schedule allows operations, such as configuration import/export or tech support collection, to occur during one or more specified windows of time.

A schedule contains a set of time windows (occurrences). These windows can be one time only or can recur at a specified time and day each week. The options defined in the window, such as the duration or the maximum number of tasks to be run, determine when a scheduled task will execute. For example, if a change cannot be deployed during a given maintenance window because the maximum duration or number of tasks has been reached, that deployment is carried over to the next maintenance window.

Each schedule checks periodically to see whether the APIC has entered one or more maintenance windows. If it has, the schedule executes the deployments that are eligible according to the constraints specified in the maintenance policy.

A schedule contains one or more occurrences, which determine the maintenance windows associated with that schedule. An occurrence can be one of the following:

- **Absolute (One Time) Window**—An absolute window defines a schedule that will occur only once. This window continues until the maximum duration of the window or the maximum number of tasks that can be run in the window has been reached.
- **Recurring Window**—A recurring window defines a repeating schedule. This window continues until the maximum number of tasks or the end of the day specified in the window has been reached.

SUMMARY STEPS

1. **configure**
2. **[no] scheduler** *schedule-name*
3. **[no] description** *text*
4. **[no] absolute window** *window-name*
5. **[no] max concurrent nodes** *count*
6. **[no] max running time** *time*
7. **[no] time start** *time*
8. **exit**
9. **[no] recurring window** *window-name*
10. **[no] max concurrent nodes** *count*
11. **[no] max running time** *time*
12. **[no] time start** {**daily** *HH:MM* | **weekly** (See usage) *HH:MM*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] scheduler <i>schedule-name</i> Example:	Creates a new scheduler or configures an existing scheduler.

	Command or Action	Purpose
	<code>apic1(config)# scheduler controller schedule myScheduler</code>	
Step 3	[no] description <i>text</i> Example: <pre>apic1(config-scheduler)# description 'This is my scheduler'</pre>	Adds a description for this scheduler. If the text includes spaces, it must be enclosed in single quotes.
Step 4	[no] absolute window <i>window-name</i> Example: <pre>apic1(config-scheduler)# absolute window myAbsoluteWindow</pre>	Creates an absolute (one time) window schedule.
Step 5	[no] max concurrent nodes <i>count</i> Example: <pre>apic1(config-scheduler-absolute)# max concurrent nodes 300</pre>	Sets the maximum number of nodes (tasks) that can be processed concurrently. The range is 0 to 65535. Set to 0 for unlimited nodes.
Step 6	[no] max running time <i>time</i> Example: <pre>apic1(config-scheduler-absolute)# max running time 00:01:30:00</pre>	Sets the maximum running time for tasks in the format dd:hh:mm:ss. The range is 0 to 65535. Set to 0 for no time limit.
Step 7	[no] time start <i>time</i> Example: <pre>apic1(config-scheduler-absolute)# time start 2016:jan:01:12:01</pre>	Sets the starting time in the format [[[:yyyy:]mmm:]dd:]HH:MM.
Step 8	exit Example: <pre>apic1(config-scheduler-absolute)# exit</pre>	Returns to scheduler configuration mode.
Step 9	[no] recurring window <i>window-name</i> Example: <pre>apic1(config-scheduler)# recurring window myRecurringWindow</pre>	Creates a recurring window schedule.
Step 10	[no] max concurrent nodes <i>count</i> Example: <pre>apic1(config-scheduler-recurring)# max concurrent nodes 300</pre>	Sets the maximum number of nodes (tasks) that can be processed concurrently. The range is 0 to 65535. Set to 0 for unlimited nodes.
Step 11	[no] max running time <i>time</i> Example: <pre>apic1(config-scheduler-recurring)# max running time 00:01:30:00</pre>	Sets the maximum running time for tasks in the format dd:hh:mm:ss. The range is 0 to 65535. Set to 0 for no time limit.

	Command or Action	Purpose
Step 12	[no] time start {daily <i>HH:MM</i> weekly (See usage) <i>HH:MM</i>} Example: <pre>apic1(config-scheduler-recurring)# time start weekly wednesday 12:30</pre>	Sets the period (daily or weekly) and starting time. If weekly is selected, choose from these options: • monday • tuesday • wednesday • thursday • friday • saturday • sunday • even-day • odd-day • every-day

Examples

This example shows how to configure a recurring scheduler to run every Wednesday.

```
apic1# configure
apic1(config)# scheduler controller schedule myScheduler
apic1(config-scheduler)# description 'This is my scheduler'
apic1(config-scheduler)# recurring window myRecurringWindow
apic1(config-scheduler-recurring)# max concurrent nodes 300
apic1(config-scheduler-recurring)# max running time 00:01:30:00
apic1(config-scheduler-recurring)# time start weekly wednesday 12:30
```

Configuring System MTU

SUMMARY STEPS

1. **configure**
2. **[no] system jumbomtu *size***

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <pre>apic1# configure</pre>	Enters global configuration mode.

	Command or Action	Purpose
Step 2	[no] system jumbomtu <i>size</i> Example: apic1(config)# system jumbomtu 9000	Sets the maximum transmit unit (MTU) for host facing ports. Up to Cisco APIC Release 3.1(2), the range is 576 to 9000 bytes. From release 3.1(2), and later, the maximum MTU value is 9216. The default has not changed from 9000.

Examples

This example shows how to configure the system MTU size.

```
apic1# configure terminal
apic1(config)# system jumbomtu 9000
```

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.

Command	Purpose
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.

Guidelines and Limitations

Follow these guidelines and limitations:

- Latency requires all the nodes in the fabric to be synchronized using Precision Time Protocol (PTP).
- Latency measurement and PTP are only supported on the following switches:
 - N9K-C93108TC-EX
 - N9K-C93108TC-FX
 - N9K-C93180LC-EX
 - N9K-C93180YC-EX
 - N9K-C93180YC-FX
 - N9K-C9364C
 - N9K-X9732C-EX
 - N9K-X9736C-EX
 - N9K-X9736C-FX
- Latency measurement is supported only for the packets that ingress, egress, and transit through EX or FX-based TORs.
- All the spine nodes in the fabric should have EX or FX-based line cards to support PTP.
- PTP and the latency feature is not supported on any N9K-C93128TX, N9K-C9396PX, and N9K-C9396TX TORs or spine switches. In the presence of non-EX/FX TORs in the fabric, we recommend that you have the external GM connectivity to all the spine switches to ensure that the PTP time is synced across all the supported TORs.
- External Grandmaster (GM) clock is not mandatory for PTP in a single Pod. If there is no external GM connected to the ACI fabric, one of the spine nodes acts as the GM. This spine switch has a PTP priority1 value as 254. All the other spine switches and leaf switches in the fabric will synchronize their clock to this Master spine switch clock. If the external GM is connected later to the spine switch, it should have a priority value less than 254 for it to act as the GM for the entire fabric.

- External Grandmaster clock is mandatory for PTP in a multipod scenario. In addition, external GM needs to be connected to the IPN such that the Grandmaster clock is the master to the spine switches in different PODs. The spine switches connected to IPN will act as the boundary clock and all the nodes within the POD will sync their clock this spine switch.
- PTP operates only in boundary clock mode. End-to-end transparent clock and peer-to-peer transparent clock modes are not supported.
- PTP supports transport over User Datagram Protocol (UDP). Transport over Ethernet is not supported.
- PTP supports multicast communication only; unicast mode is not supported.
- Beginning with release 4.0(1), support is added for changing the resolution factor to 11 which then can measure up to 214 milliseconds with an accuracy of 204ns.

Configuring PTP Using the NX-OS CLI

Step 1 Enable PTP.

Example:

```
Enable ptp:
=====
apic# configure terminal
apic(config)# ptp
Disable ptp:
=====
apic# configure terminal
apic(config)# no ptp
```

Step 2 To verify PTP on ACI switches:

Example:

```
leaf1# show ptp brief
PTP port status
-----
Port          State
-----
Eth1/49       Slave

leaf1#
leaf1#
leaf1# show ptp clock
PTP Device Type: Boundary clock
Clock Identity : 0c:75:bd:ff:fe:03:1d:10
Clock Domain: 0
Number of PTP ports: 1
Priority1 : 255
Priority2 : 255
Clock Quality:
    Class : 248
    Accuracy : 254
    Offset (log variance) : 65535
Offset From Master : 32
Mean Path Delay : 128
Steps removed : 1
Local clock time:Thu Jul 27 19:43:42 2017
```

```
leaf1#
leaf1# show ptp clock foreign-masters record interface ethernet 1/49
```

P1=Priority1, P2=Priority2, C=Class, A=Accuracy,
OSLV=Offset-Scaled-Log-Variance, SR=Steps-Removed
GM=Is grandmaster

Interface	Clock-ID	P1	P2	C	A	OSLV	SR
Eth1/49	d4:6d:50:ff:fe:e6:4d:3f	254	255	248	254	65535	0 GM

```
leaf1#
leaf1#
leaf1# show ptp corrections
```

PTP past corrections

Slave Port	SUP Time	Correction(ns)	MeanPath Delay(ns)
Eth1/49	Thu Jul 27 19:44:11 2017 364281	36	152
Eth1/49	Thu Jul 27 19:44:11 2017 114565	16	132
Eth1/49	Thu Jul 27 19:44:10 2017 862912	8	132
Eth1/49	Thu Jul 27 19:44:10 2017 610823	8	132
Eth1/49	Thu Jul 27 19:44:10 2017 359557	16	132
Eth1/49	Thu Jul 27 19:44:10 2017 109937	8	132
Eth1/49	Thu Jul 27 19:44:09 2017 858113	16	132
Eth1/49	Thu Jul 27 19:44:09 2017 606536	16	132
Eth1/49	Thu Jul 27 19:44:09 2017 354837	-16	132
Eth1/49	Thu Jul 27 19:44:09 2017 104226	24	148
Eth1/49	Thu Jul 27 19:44:08 2017 853263	24	148
Eth1/49	Thu Jul 27 19:44:08 2017 601780	16	148
Eth1/49	Thu Jul 27 19:44:08 2017 349639	-4	148
Eth1/49	Thu Jul 27 19:44:08 2017 99970	16	144
Eth1/49	Thu Jul 27 19:44:07 2017 848507	0	144
Eth1/49	Thu Jul 27 19:44:07 2017 596143	24	144
Eth1/49	Thu Jul 27 19:44:07 2017 344808	4	144
Eth1/49	Thu Jul 27 19:44:07 2017 93156	-16	140
Eth1/49	Thu Jul 27 19:44:06 2017 843263	24	140
Eth1/49	Thu Jul 27 19:44:06 2017 590189	8	140

```
leaf1#
leaf1#
leaf1# show ptp counters all
```

PTP Packet Counters of Interface Eth1/49:

Packet Type	TX	RX
Announce	56	5424
Sync	441	43322
FollowUp	441	43321
Delay Request	7002	0
Delay Response	0	7002
PDelay Request	0	0
PDelay Response	0	0
PDelay Followup	0	0
Management	0	0

```
leaf1#
leaf1#
leaf1# show ptp parent
```

PTP PARENT PROPERTIES

Parent Clock:

Parent Clock Identity: d4:6d:50:ff:fe:e6:4d:3f

Parent Port Number: 258

Observed Parent Offset (log variance): N/A

Observed Parent Clock Phase Change Rate: N/A

Grandmaster Clock:

Grandmaster Clock Identity: d4:6d:50:ff:fe:e6:4d:3f

Grandmaster Clock Quality:

Class: 248

Accuracy: 254

Offset (log variance): 65535

Priority1: 254

Priority2: 255

leaf1#

Step 3

To verify troubleshooting steps:

Example:

apic1# show troubleshoot eptoe session eptoe latency

Source --> Destination

Last Collection(30 seconds)

Average (microsec)	Standard Deviation (microsec)	Packet Count
18	24	1086

Cumulative

Average (microsec)	Max (microsec)	Total Packet Count
18	202	6117438



CHAPTER 15

Configuring Cisco Tetration Analytics

- [Overview, on page 461](#)
- [Configuring Cisco Tetration Analytics Using the NX-OS Style CLI, on page 461](#)

Overview

This article provides examples of how to configure Cisco Tetration Analytics when using the Cisco APIC. The following information applies when configuring Cisco Tetration Analytics.

- An inband management IP address must be configured on each leaf where the Cisco Tetration Analytics agent is active.
- Define an analytics policy and specify the destination IP address of the Cisco Tetration Analytics server.
- Create a switch profile and include the policy group created in the previous step.

Configuring Cisco Tetration Analytics Using the NX-OS Style CLI

Step 1 **configure terminal**

Enters global configuration mode.

Example:

```
apic1# configure terminal
```

Step 2 **analytics cluster *cluster_name***

Create the analytics policy.

Example:

```
apic1(config)# analytics cluster cluster1
```

Step 3 **flow-exporter *server_name***

Configure external analytics information.

Example:

```
apic1(config-analytics)# flow-exporter server1
```

Step 4

destination *ip_address*

Configure the destination port.

Example:

```
apic1(config-analytics-cluster-exporter)# destination 192.0.2.1
```

Step 5

exit

Exit command mode.

Example:

```
# apic1(config-analytics-cluster-exporter)# exit
```

Step 6

exit

Exit command mode.

Example:

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

Step 7

fabric-internal

Enters fabric internal configuration mode.

Example:

```
apic1(config)# fabric-internal
```

Step 8

template leaf-policy-group *leaf_group_name*

Define leaf policy group.

Example:

```
apic1(config-fabric-internal)# template leaf-policy-group lpg1
```

Step 9

inherit analytics-policy cluster *cluster_name* **server** *server_name*

Associate analytics policy to leaf policy group.

Example:

```
apic1(config-leaf-policy-group)# inherit analytics-policy cluster cluster1 server server1
```

Step 10

exit

Exit command mode.

Example:

```
apic1(config-leaf-policy-group)# exit
```

Step 11

leaf-profile *leaf_profile_name*

Define leaf profile.

Example:

```
apic1(config-fabric-internal)# leaf-profile lp1
```

Step 12

leaf-group *leaf_group_name*

Define leaf group.

Example:

```
apic1(config-leaf-profile)# leaf-group lg1
```

Step 13 **leaf-policy-group** *leaf_policy_group_name*

Associate leaf policy group to leaf group.

Example:

```
apic1(config-leaf-group)# leaf-policy-group lpg1
```

Step 14 **leaf** *leaf_group_number*

Add nodes to leaf group.

Example:

```
apic1(config-leaf-group)# leaf 101
```

Step 15 **show analytics**

Display analytics.

Note The destination port is not configurable. UDP port 5640 is always used for leaf switches and UDP port 5641 is always used for spine switches.

The DSCP is not configurable. VA (Voice Admit) is always used.

Example:

```
apic1# show analytics
Cluster                : cluster1
Config Server Name    : server1
Destination IP        : 192.0.2.1
Destination Port      : unspecified
DSCP                  : VA
```

Step 16 **show running-config analytics**

Display running configuration analytics.

Example:

```
apic1# show running-config analytics
# Command: show running-config analytics
# Time: Wed May 25 21:14:43 2016
analytics cluster cluster1
  flow-exporter server1
    destination 192.0.2.1
    destination-port unspecified
    dscp VA
    ip-filter-action deny
  exit
exit
apic1#
```




CHAPTER 16

Configuring NetFlow

- [About NetFlow, on page 465](#)
- [Configuring a NetFlow Exporter Policy for Virtual Machine Networking Using the NX-OS-Style CLI, on page 466](#)
- [Configuring NetFlow and Tetration Analytics Feature Priority Through Node Control Policy Using NX-OS-Style CLI, on page 466](#)
- [Configuring NetFlow Node Policy Using the NX-OS-Style CLI, on page 467](#)
- [Configuring NetFlow Infra Selectors Using the NX-OS-Style CLI, on page 467](#)
- [Configuring NetFlow Overrides Using the NX-OS-Style CLI, on page 469](#)
- [Configuring NetFlow Tenant Hierarchy Using the NX-OS-Style CLI, on page 470](#)
- [Consuming a NetFlow Exporter Policy Under a VMM Domain Using the NX-OS-Style CLI for VMware VDS, on page 473](#)
- [Enabling or Disabling NetFlow on an Endpoint Group Using the NX-OS-Style CLI for VMware VDS, on page 474](#)

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 information about configuring NetFlow with virtual machine networking, see the *Cisco ACI Virtualization Guide*.

Configuring a NetFlow Exporter Policy for Virtual Machine Networking Using the NX-OS-Style CLI

The following example procedure uses the NX-OS-style CLI to configure a NetFlow exporter policy for virtual machine networking.

Step 1 Enter the configuration mode.

Example:

```
apic1# config
```

Step 2 Configure the exporter policy.

Example:

```
apic1(config)# flow vm-exporter vmExporter1 destination address 2.2.2.2 transport udp 1234
apic1(config-flow-vm-exporter)# source address 4.4.4.4
apic1(config-flow-vm-exporter)# exit
apic1(config)# exit
```

Configuring NetFlow and Tetration Analytics Feature Priority Through Node Control Policy Using NX-OS-Style CLI

The following example procedure uses the NX-OS-style CLI to configure the NetFlow and Tetration Analytics feature priority through a node control policy:

Step 1 Enter the configuration mode.

Example:

```
apic1# config
```

Step 2 Create a node control policy.

Example:

```
apic1(config)# node-control policy poll
```

Step 3 Set NetFlow as the priority feature.

Example:

```
apic1(config-node)# feature netflow
```

Step 4 Exit the node control policy configuration.

Example:

```
apic1(config-node)# end
```

Step 5 Deploy the policy to node 101 and node 102.

Example:

```
ifav-isim15-ifc1(config)# fabric-internal
ifav-isim15-ifc1(config-fabric-internal)# template leaf-policy-group lpg1
ifav-isim15-ifc1(config-leaf-policy-group)# inherit node-control-policy poll
ifav-isim15-ifc1(config-leaf-policy-group)# exit
ifav-isim15-ifc1(config-fabric-internal)# leaf-profile leafProfile1
ifav-isim15-ifc1(config-leaf-profile)# leaf-group leafgrp1
ifav-isim15-ifc1(config-leaf-group)# leaf 101
ifav-isim15-ifc1(config-leaf-group)# leaf 102
ifav-isim15-ifc1(config-leaf-group)# leaf-policy-group lpg1
ifav-isim15-ifc1(config-leaf-group)# end
```

Configuring NetFlow Node Policy Using the NX-OS-Style CLI

The following example procedure uses the NX-OS-style CLI to configure a NetFlow node policy:

Step 1 Enter the configuration mode.

Example:

```
apic1# config
```

Step 2 Configure the node policy.

Example:

```
apic1(config)# flow node-policy nodePol
apic1(config-flow-node-pol)# flow timeout collection 100
apic1(config-flow-node-pol)# flow timeout template 123
apic1(config-flow-node-pol)# exit
```

Configuring NetFlow Infra Selectors Using the NX-OS-Style CLI

You can use the NX-OS-style CLI 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 CLI commands show how to configure NetFlow infra selectors using the NX-OS-style CLI:

Step 1 Enter the configuration mode.

Example:

```
apic1# config
```

Step 2 Create a NetFlow exporter policy.

Example:

In the following commands, the destination endpoint group is the endpoint group that the exporter sits behind. This endpoint group can also be an external Layer 3 endpoint group.

```
apic1(config)# flow exporter infraExporter1 destination address 1.2.3.4 transpo udp 1234
apic1(config-flow-exporter)# destination epg tenant tn2 application ap2 epg epg2
apic1(config-flow-exporter)# vrf member tenant tn2 vrf vrf2
apic1(config-flow-exporter)# version v9
apic1(config-flow-exporter)# source address 1.1.1.1
apic1(config-flow-exporter)# exit
```

Step 3 Create a second NetFlow exporter policy.

Example:

In the following commands, the destination endpoint group is the endpoint group that the exporter sits behind, which in this case is an external Layer 3 endpoint group.

```
apic1(config)# flow exporter infraExporter2
apic1(config-flow-exporter)# transport udp 9990
apic1(config-flow-exporter)# destination address 2001:db5:a0c:1f0::2
apic1(config-flow-exporter)# destination external-l3 epg tenant tn2 vrf v2 epg accounting-inst
apic1(config-flow-exporter)# vrf member tenant tn2 vrf vrf2
apic1(config-flow-exporter)# version v5
apic1(config-flow-exporter)# source address 2001:db8:a0b:12f0::1
apic1(config-flow-exporter)# exit
```

Step 4 Create a NetFlow record policy.

Example:

```
apic1(config)# flow record infraRecord1
apic1(config-flow-record)# match dst-ip
apic1(config-flow-record)# match dst-ipv4
apic1(config-flow-record)# match dst-ipv6
apic1(config-flow-record)# match dst-mac
apic1(config-flow-record)# match dst-port
apic1(config-flow-record)# match ethertype
apic1(config-flow-record)# match proto
apic1(config-flow-record)# match src-ip
apic1(config-flow-record)# match src-ipv4
apic1(config-flow-record)# match src-ipv6
apic1(config-flow-record)# match src-mac
apic1(config-flow-record)# match src-port
apic1(config-flow-record)# match tos
apic1(config-flow-record)# match vlan
apic1(config-flow-record)# collect count-bytes
apic1(config-flow-record)# collect count-pkts
apic1(config-flow-record)# collect pkt-disp
apic1(config-flow-record)# collect sampler-id
apic1(config-flow-record)# collect src-intf
apic1(config-flow-record)# collect tcp-flags
apic1(config-flow-record)# collect ts-first
apic1(config-flow-record)# collect ts-recent
apic1(config-flow-record)# exit
```

Step 5 Create a NetFlow monitor policy.

Example:

```
apic1(config)# flow monitor infraMonitor1
apic1(config-flow-monitor)# record infraRecord1
apic1(config-flow-monitor)# exporter infraExporter1
apic1(config-flow-monitor)# exporter infraExporter2
apic1(config-flow-monitor)# exit
```


You can attach a maximum of two exporters.

Step 6 Create an interface policy group (AccPortGrp).

Example:

```
apic1(config)# template policy-group pg1
apic1(config-pol-grp-if)# ip flow monitor infraMonitor1
apic1(config-pol-grp-if)# ipv6 flow monitor infraMonitor2
apic1(config-pol-grp-if)# exit
```

You can have one monitor policy per address family (IPv4 and IPv6).

Step 7 Create a node profile and infra selectors.

Example:

```
apic1(config)# leaf-profile lp1
apic1(config-leaf-profile)# leaf-group lg1
apic1(config-leaf-group)# leaf 101
apic1(config-leaf-profile)# exit
apic1(config)# leaf-interface-profile lip1
apic1(config-leaf-if-profile)# exit
apic1(config)# leaf-interface-profile lip1
apic1(config-leaf-if-profile)# leaf-interface-group lig1
apic1(config-leaf-if-group)# interface ethernet 1/5
apic1(config-leaf-if-profile)# policy-group pg1
apic1(config-leaf-if-profile)# exit
apic1(config-leaf-profile)# exit
```

Step 8 Create a port channel policy group (AccBndlGrp).

Example:

```
apic1(config)# template port-channel po6
apic1(config-if)# ip flow monitor infraMonitor1
apic1(config-if)# ipv6 flow monitor infraMonitor1
apic1(config-if)# exit
apic1(config-leaf-profile)# leaf-profile lp2
apic1(config-leaf-group)# leaf-group lg2
apic1(config-leaf-profile)# leaf 101
apic1(config-leaf-profile)# exit
apic1(config)# leaf-interface-profile lip2
apic1(config-leaf-if-profile)# exit
apic1(config)# leaf-interface-profile lip2
apic1(config-leaf-if-profile)# leaf-interface-group lig2
apic1(config-leaf-if-group)# interface ethernet 1/6
apic1(config-leaf-if-profile)# channel-group po6
apic1(config-leaf-if-profile)# exit
```

You can have one monitor policy per address family (IPv4 and IPv6). The interfaces can also be vPCs.

Configuring NetFlow Overrides Using the NX-OS-Style CLI

The following procedure configures NetFlow overrides using the NX-OS-Style CLI:

Step 1 Enter the configuration mode.

Example:

```
apic1# config
```

Step 2 Create the override.

Example:

```
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant tn2 vrf vrf2
apic1(config-leaf)# exit
apic1(config)# interface ethernet 1/15
apic1(config-if)# ip flow monitor infraMonitor1
apic1(config-if)# ipv6 flow monitor infraMonitor2
apic1(config-if)# exit
apic1(config)# exit
apic1# exit
```

You can have one monitor policy per address family (IPv4 and IPv6). The interfaces can also be vPCs.

Configuring NetFlow Tenant Hierarchy Using the NX-OS-Style CLI

The following example procedure uses the NX-OS-style CLI to configure the NetFlow tenant hierarchy:

Step 1 Enter the configuration mode.

Example:

```
apic1# config
```

Step 2 Create a tenant and bridge domain, and add them to a VRF.

Example:

```
apic1(config)# tenant tn2
apic1(config-tenant)# vrf context vrf2
apic1(config-tenant-vrf)# exit
apic1(config-tenant)# bridge-domain bd2
apic1(config-tenant-bridge-domain)# vrf member vrf2
apic1(config-tenant-bridge-domain)# exit
apic1(config-tenant)# bridge-domain bd3
apic1(config-tenant-bridge-domain)# vrf member vrf2
apic1(config-tenant-bridge-domain)# exit
```

Step 3 Create an application endpoint group behind which the exporter resides.

Example:

```
apic1(config-tenant)# application ap2
apic1(config-tenant-app)# epg epg2
apic1(config-tenant-app)# bridge-domain member bd2
apic1(config-tenant-app-bridge-domain)# exit
apic1(config-tenant-app)# exit
```

Step 4 Create a second application endpoint group behind which the exporter resides.

Example:

```
apic1(config-tenant)# application ap3
apic1(config-tenant-app)# epg epg3
apic1(config-tenant-app)# bridge-domain member bd3
apic1(config-tenant-app-bridge-domain)# exit
apic1(config-tenant-app)# exit
```

Step 5 Attach a NetFlow monitor policy on the bridge domains.

Example:

```
apic1(config)# interface bridge-domain bd2
apic1(config-if)# ipv6 flow monitor tnMonitor1
apic1(config-if)# ip flow monitor tnMonitor1
apic1(config-if)# layer2-switched flow monitor tnMonitor1
apic1(config-if)# exit
apic1(config)# interface bridge-domain bd3
apic1(config-if)# ipv6 flow monitor tnMonitor1
apic1(config-if)# ip flow monitor tnMonitor1
apic1(config-if)# exit
```

You can have one monitor policy per address family (IPv4 and IPv6). The interfaces can also be vPCs.

Step 6 Create the Netflow exporter policy.

Example:

In the following commands, the destination endpoint group is the endpoint group that the exporter sits behind. This endpoint group can also be an external Layer 3 endpoint group.

```
apic1(config)# flow exporter tnExporter1
apic1(config-flow-exporter)# transport udp 1234
apic1(config-flow-exporter)# destination address 2.2.2.2
apic1(config-flow-exporter)# destination epg tenant tn2 application ap2 epg epg2
apic1(config-flow-exporter)# vrf member tenant tn2 vrf vrf2
apic1(config-flow-exporter)# version v9
apic1(config-flow-exporter)# source address 1.1.1.1
apic1(config-flow-exporter)# exit
```

Step 7 Create a second Netflow exporter policy.

Example:

In the following commands, the destination endpoint group is the endpoint group that the exporter sits behind, which in this case is an external Layer 3 endpoint group.

```
apic1(config)# flow exporter tnExporter2
apic1(config-flow-exporter)# transport udp 9990
apic1(config-flow-exporter)# destination address 2001:db5:a0c:1f0::2
apic1(config-flow-exporter)# destination external-l3 epg tenant tn2 vrf v2 epg accounting-inst
apic1(config-flow-exporter)# vrf member tenant tn2 vrf vrf2
apic1(config-flow-exporter)# version v5
apic1(config-flow-exporter)# source address 2001:db8:a0b:12f0::1
apic1(config-flow-exporter)# exit
```

Step 8 Create a NetFlow record policy.

Example:

```
apic1(config)# flow record tnRecord1
apic1(config-flow-record)# match dst-ip
apic1(config-flow-record)# match dst-ipv4
apic1(config-flow-record)# match dst-ipv6
apic1(config-flow-record)# match dst-mac
```

```

apic1(config-flow-record)# match dst-port
apic1(config-flow-record)# match ethertype
apic1(config-flow-record)# match proto
apic1(config-flow-record)# match src-ip
apic1(config-flow-record)# match src-ipv4
apic1(config-flow-record)# match src-ipv6
apic1(config-flow-record)# match src-mac
apic1(config-flow-record)# match src-port
apic1(config-flow-record)# match tos
apic1(config-flow-record)# match vlan
apic1(config-flow-record)# collect count-bytes
apic1(config-flow-record)# collect count-pkts
apic1(config-flow-record)# collect pkt-disp
apic1(config-flow-record)# collect sampler-id
apic1(config-flow-record)# collect src-intf
apic1(config-flow-record)# collect tcp-flags
apic1(config-flow-record)# collect ts-first
apic1(config-flow-record)# collect ts-recent
apic1(config-flow-record)# exit

```

Step 9 Create a NetFlow monitor policy.

Example:

```

apic1(config)# flow monitor tnMonitor1
apic1(config-flow-monitor)# record tnRecord1
apic1(config-flow-monitor)# exporter tnExporter1
apic1(config-flow-monitor)# exporter tnExporter2
apic1(config-flow-monitor)# exit

```

You can attach a maximum of two exporters.

Step 10 Add VLANs to the VLAN domain and configure a VRF for a leaf node.

Example:

```

apic1(config)# vlan-domain dom1
apic1(config-vlan)# vlan 5-100
apic1(config-vlan)# exit
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant tn2 vrf vrf2
apic1(config-leaf-vrf)# exit

```

Step 11 Deploy an endpoint group on an interface to deploy the bridge domain.

Example:

```

apic1(config-leaf)# interface ethernet 1/10
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk allowed vlan 10 tenant tn2 application ap2 ep2 ep2
apic1(config-leaf-if)# exit

```

Step 12 Deploy another endpoint group on an interface.

Example:

```

apic1(config-leaf)# interface ethernet 1/11
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk allowed vlan 11 tenant tn2 application ap3 ep3 ep3
apic1(config-leaf-if)# exit

```

Step 13 Attach the monitor policy to the sub-interface.

Example:

```

apic1(config-leaf)# interface ethernet 1/20
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 1/20.20
apic1(config-leaf-if)# vrf member tenant tn2 vrf vrf2
apic1(config-leaf-if)# ipv6 address 20::1/64 preferred
apic1(config-leaf-if)# ipv6 flow monitor tnMonitor1
apic1(config-leaf-if)# ip flow monitor tnMonitor2
apic1(config-leaf-if)# exit

```

Step 14 Attach the monitor policy to a switched virtual interface (SVI).

Example:

```

apic1(config-leaf)# interface vlan 30
apic1(config-leaf-if)# vrf member tenant tn2 vrf vrf2
apic1(config-leaf-if)# ipv6 address 64::1/64 preferred
apic1(config-leaf-if)# ip flow monitor tnMonitor1
apic1(config-leaf-if)# ip6 flow monitor tnMonitor1
apic1(config-leaf-if)# exit

```

Step 15 Associate the SVI to a Layer 2 interface.

Example:

```

apic1(config-leaf)# interface ethernet 1/30
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk allowed vlan 30 tenant tn2 external-svi
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# exit

```

Consuming a NetFlow Exporter Policy Under a VMM Domain Using the NX-OS-Style CLI for VMware VDS

The following procedure uses the NX-OS-style CLI to consume a NetFlow exporter policy under a VMM domain.

Step 1 Enter the configuration mode.

Example:

```
apic1# config
```

Step 2 Consume the NetFlow exporter policy.

Example:

```

apic1(config)# vmware-domain mininet
apic1(config-vmware)# configure-dvs
apic1(config-vmware-dvs)# flow exporter vmExporter1
apic1(config-vmware-dvs-flow-exporter)# active-flow-timeout 62
apic1(config-vmware-dvs-flow-exporter)# idle-flow-timeout 16
apic1(config-vmware-dvs-flow-exporter)# sampling-rate 1
apic1(config-vmware-dvs-flow-exporter)# exit

```

```
apic1(config-vmware-dvs)# exit
apic1(config-vmware)# exit
apic1(config)# exit
```

Enabling or Disabling NetFlow on an Endpoint Group Using the NX-OS-Style CLI for VMware VDS

The following procedure enables or disables NetFlow on an endpoint group using the NX-OS-style CLI.

Step 1 Enable NetFlow:

Example:

```
apic1# config
apic1(config)# tenant tn1
apic1(config-tenant)# application appl
apic1(config-tenant-app)# epg epg1
apic1(config-tenant-app-epg)# vmware-domain member mininet
apic1(config-tenant-app-epg-domain)# flow monitor enable
apic1(config-tenant-app-epg-domain)# exit
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)# exit
apic1(config-tenant)# exit
apic1(config)# exit
```

Step 2 (Optional) If you no longer want to use NetFlow, disable the feature:

Example:

```
apic1(config-tenant-app-epg-domain)# no flow monitor enable
```



CHAPTER 17

Managing Firmware

- [Managing Firmware, on page 475](#)
- [Adding or Removing Repository Images, on page 475](#)
- [Changing Catalog Firmware, on page 476](#)
- [Upgrading Controller Firmware, on page 477](#)
- [Upgrading Switch Firmware, on page 479](#)

Managing Firmware

Each firmware image includes a compatibility catalog that identifies supported types and switch models. APIC maintains a catalog of the firmware images, switch types, and models that are allowed to use that firmware image. The default setting is to reject a firmware update when it does not conform to the compatibility catalog.

APIC has an image repository for compatibility catalogs, controller firmware images, and switch images. The administrator can download new firmware image to the APIC image repository from an external HTTP server or SCP server.



Note Before you upgrade the switches, the APICs must have completed upgrading and have a health state of Fully Fit.

Adding or Removing Repository Images

SUMMARY STEPS

1. `firmware repository add absolute-image-path`
2. `firmware repository delete image`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>firmware repository add <i>absolute-image-path</i></code> Example:	Adds a firmware image to the repository.

	Command or Action	Purpose
	<pre>apicl# firmware repository add /home/admin/aci-catalog-dk9.1.2.1b.bin</pre>	
Step 2	firmware repository delete <i>image</i> Example: <pre>apicl# firmware repository delete aci-catalog-dk9.1.2.1a.bin</pre>	Deletes a firmware image from the repository.

Examples

```
apicl# firmware repository add /home/admin/aci-catalog-dk9.1.2.1b.bin
apicl# firmware repository delete aci-catalog-dk9.1.2.1a.bin
```

Changing Catalog Firmware

This procedure shows how to select a catalog firmware version from the repository.

SUMMARY STEPS

1. **show** firmware repository [detail]
2. **configure**
3. **firmware**
4. (Optional) **show version**
5. **catalog-version** *firmware-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	show firmware repository [detail] Example: <pre>apicl# show firmware repository</pre>	Show firmware images present in repository. The detail option displays additional information such as MD5 checksum, release date, and download date.
Step 2	configure Example: <pre>apicl# configure</pre>	Enters global configuration mode.
Step 3	firmware Example: <pre>apicl(config)# firmware</pre>	Enters firmware upgrade configuration mode.
Step 4	(Optional) show version Example:	Displays the currently-installed controller and switch firmware versions.

	Command or Action	Purpose
	<code>apic1(config-firmware)# show version</code>	
Step 5	catalog-version <i>firmware-name</i> Example: <code>apic1(config-firmware)# catalog-version</code> <code>aci-catalog-dk9.1.2.1b.bin</code>	Changes the catalog version to an available image in the repository.

Examples

This example shows how to select a catalog firmware version from the repository.

```
apic1# show firmware repository
Name                               Type      Version  Size (MB)
-----
aci-catalog-dk9.1.2.1a.bin         catalog   1.2.1a   0.023
aci-catalog-dk9.1.2.1b.bin         catalog   1.2.1b   0.025

apic1# configure
apic1(config)# firmware
apic1(config-firmware)# catalog-version aci-catalog-dk9.1.2.1b.bin
```

Upgrading Controller Firmware

The controllers upgrade in random order. Each APIC controller takes about 10 minutes to upgrade. Once a controller image is upgraded, it drops from the cluster and reboots with the newer version while the other APIC controllers in the cluster are still operational. Once the controller reboots, it joins the cluster again. Then the cluster converges, and the next controller image starts to upgrade.

The catalog firmware image is upgraded when an APIC controller image is upgraded. You do not need to upgrade the catalog firmware image separately.

SUMMARY STEPS

1. **configure**
2. **firmware**
3. (Optional) **show version**
4. **controller-group**
5. **firmware-version** *firmware-name*
6. **[no] time start** *time*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.

	Command or Action	Purpose
Step 2	firmware Example: <code>apicl(config)# firmware</code>	Enters firmware upgrade configuration mode.
Step 3	(Optional) show version Example: <code>apicl(config-firmware)# show version</code>	Displays the currently-installed controller and switch firmware versions.
Step 4	controller-group Example: <code>apicl(config-firmware)# controller-group</code>	Enters controller upgrade configuration mode.
Step 5	firmware-version <i>firmware-name</i> Example: <code>apicl(config-firmware-controller)# firmware-version aci-apic-dk9.1.2.1b.iso</code>	Specifies the desired version for the upgrade.
Step 6	[no] time start <i>time</i> Example: <code>apicl(config-firmware-controller)# time start 2016:jan:01:12:01</code>	Sets the starting time in the format <code>[[[yyyy:]mmm:]dd:]HH:MM</code> . The date is optional. Note To upgrade the controllers immediately, return to EXEC mode and type the command firmware upgrade controller-group .

Examples

This example shows how to upgrade the controllers.

```
apicl# show controller
Fabric Name       : mininet
Operational Size  : 3
Cluster Size      : 3
Time Difference    : 0
Fabric Security Mode : permissive
```

ID	Address	In-Band Address	OOB Address	Version	Flags	Serial Number	Health
1*	10.0.0.1	192.168.11.1	192.168.10.1	1.2(1a)	crva	TEP-1-1	fully-fit
2	10.0.0.2	192.168.11.2	192.168.10.2	1.2(1a)	crva	TEP-1-2	fully-fit
3	10.0.0.3	192.168.11.3	192.168.10.3	1.2(1a)	crva	TEP-1-3	fully-fit

Flags - c:Commissioned | r:Registered | v:Valid Certificate | a:Approved

```
apicl# configure
apicl(config)# firmware
apicl(config-firmware)# show version
```

Role	Id	Name	Version
controller	1	apic1	1.2 (1a)
controller	2	apic2	1.2 (1a)
controller	3	apic3	1.2 (1a)
leaf	101	leaf1	n9000-11.2 (1a)
leaf	102	leaf2	n9000-11.2 (1a)
leaf	103	leaf2	n9000-11.2 (1a)
spine	201	spine1	n9000-11.2 (1a)
spine	202	spine2	n9000-11.2 (1a)

```

apic1(config-firmware)# controller-group
apic1(config-firmware-controller)# firmware-version aci-apic-dk9.1.2.1b.iso
apic1(config-firmware-controller)# time start 2016:jan:01:12:01

```

Upgrading Switch Firmware

Before you begin

A scheduler must exist to specify when the upgrade will be executed.



Note Before you upgrade the switches, the APICs must have completed upgrading and have a health state of Fully Fit.

SUMMARY STEPS

1. **configure**
2. **firmware**
3. **[no] switch-group** *group-name*
4. **[no] switch** *node-id-or-name[,node-id-or-name,...]*
5. **firmware-version** *firmware-name*
6. **[no] run-mode** {**pause-never** | **pause-on-failure**}
7. **schedule** *scheduler-name*
8. **[no] scheduler** **pause**
9. **show running-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	firmware Example: apic1(config)# firmware	Enters firmware upgrade configuration mode.

	Command or Action	Purpose
Step 3	[no] switch-group <i>group-name</i> Example: <code>apic1(config-firmware) # switch-group mySwitchGroup5</code>	Creates (or deletes) switch group and enters switch upgrade configuration mode.
Step 4	[no] switch <i>node-id-or-name</i> [, <i>node-id-or-name</i> ,...] Example: <code>apic1(config-firmware-switch) # switch leaf1-leaf3,leaf6</code> <code>apic1(config-firmware-switch) # no switch leaf4,leaf5</code>	Adds (or removes) a switch or a list of switches to the switch-group for upgrading. You can specify the node ID (such as 101) or the name (such as spine1). You can specify multiple switches by using commas.
Step 5	firmware-version <i>firmware-name</i> Example: <code>apic1(config-firmware-switch) # firmware-version aci-apic-dk9.11.2.1a.bin</code>	Specifies the target firmware image.
Step 6	[no] run-mode { pause-never pause-on-failure } Example: <code>apic1(config-firmware-switch) # run-mode pause-on-failure</code>	Specifies whether to proceed to the next set of nodes if the upgrade fails on the current set of nodes.
Step 7	schedule <i>scheduler-name</i> Example: <code>apic1(config-firmware-switch) # schedule myNextSunday</code>	Assigns a scheduler for the upgrade. Enter the name of a scheduler that has already been defined. Note To upgrade the switch group immediately, return to EXEC mode and type the command firmware upgrade switch-group .
Step 8	[no] scheduler pause Example: <code>apic1(config-firmware-switch) # scheduler pause</code> <code>apic1(config-firmware-switch) # no scheduler pause</code>	Pauses the maintenance policy scheduler. Use the [no] prefix to resume.
Step 9	show running-config Example: <code>apic1(config-firmware-switch) # show run</code>	Displays the configuration.

Examples

This example shows how to upgrade the firmware for three leaf switches.

```
apic1# configure
apic1(config)# firmware
apic1(config-firmware)# switch-group mySwitchGroup5
apic1(config-firmware-switch)# switch leaf1,leaf3,leaf6
apic1(config-firmware-switch)# no switch leaf4,leaf5
```

```
apic1(config-firmware-switch)# firmware-version aci-apic-dk9.1.1.3f.bin
apic1(config-firmware-switch)# run-mode pause-on-failure
apic1(config-firmware-switch)# schedule myNextSunday
apic1(config-firmware-switch)# show run
# Command: show running-config firmware switch-group mySwitchGroup5
# Time: Fri Nov 6 23:55:35 2015
firmware
  switch-group mySwitchGroup5
    switch 101
    switch 102
    switch 103
    switch 106
    schedule myNextSunday
  exit
exit
```




CHAPTER 18

Managing the Configuration with Snapshots

- [About Configuration Management and Snapshots, on page 483](#)
- [Exporting a Snapshot, on page 483](#)
- [Importing a Snapshot, on page 485](#)
- [Rollback Configuration Using Snapshots, on page 486](#)
- [Uploading or Downloading a Snapshot File to a Remote Path, on page 488](#)
- [Managing Snapshot Files and Jobs, on page 489](#)

About Configuration Management and Snapshots

You can back up and restore your system configuration by exporting and importing configuration archives (snapshots) to and from a local controller-managed folder. By exporting snapshots before and after making configuration changes, you have the ability to roll back configuration changes that were applied between two snapshots.

You can also upload and download the snapshot files to and from a remote server.

Each snapshot action (export, import, rollback, upload, and download) is performed by creating a policy for the action and then triggering the action as a job. Export actions can also be scheduled to run at a future time or periodically. Import, export, and rollback jobs cannot run in parallel. If a job is already running, triggering a new job will fail.

Exporting a Snapshot

Before you begin

If you want to export snapshots according to a schedule, configure a scheduler before configuring the export policy.

SUMMARY STEPS

1. **configure**
2. **[no] snapshot export** *policy-name*
3. **format** {xml | json}
4. (Optional) **[no] schedule** *schedule-name*
5. (Optional) **[no] target** [infra | fabric | *tenant-name*]

6. (Optional) **[no] remote path** *remote-path-name*
7. **end**
8. **trigger snapshot export** *policy-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apicl# configure</code>	Enters global configuration mode.
Step 2	[no] snapshot export <i>policy-name</i> Example: <code>apicl(config)# snapshot export myExportPolicy</code>	Creates a policy for exporting snapshots.
Step 3	format {xml json} Example: <code>apicl(config-export)# format json</code>	Specifies the data format for the exported configuration file. The default is
Step 4	(Optional) [no] schedule <i>schedule-name</i> Example: <code>apicl(config-export)# schedule EveryEightHours</code>	Specifies an existing scheduler for exporting snapshots.
Step 5	(Optional) [no] target [infra fabric tenant-name] Example: <code>apicl(config-export)# target tenantExampleCorp</code>	Assigns the target of the export, which can be fabric, infra, a specific tenant, or none. If no target is specified, all configuration information is exported. The default is no target.
Step 6	(Optional) [no] remote path <i>remote-path-name</i> Example: <code>apicl(config-export)# remote path myBackupServer</code>	Specifies the name of a configured remote path to which the file will be sent. If no remote path is specified, the file is exported locally to a folder in the controller. The default is no remote path.
Step 7	end Example: <code>apicl(config-export)# end</code>	Returns to EXEC mode.
Step 8	Required: trigger snapshot export <i>policy-name</i> Example: <code>apicl# trigger snapshot export myExportPolicy</code>	Executes the snapshot export task. If the export policy is configured with a scheduler, this step is unnecessary unless you want an immediate export.

Examples

This example shows how to configure the periodic export of a JSON-format snapshot file for a specific tenant configuration.

```
apicl# configure
```



```

apic1(config)# snapshot export myExportPolicy
apic1(config-export)# format json
apic1(config-export)# target tenantExampleCorp
apic1(config-export)# schedule EveryEightHours

```

Importing a Snapshot

SUMMARY STEPS

1. **configure**
2. **[no] snapshot import** *policy-name*
3. **file** *filename*
4. **action** {merge | replace}
5. **[no] mode** {atomic | best-effort}
6. (Optional) **[no] remote path** *remote-path-name*
7. **end**
8. **trigger snapshot import** *policy-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	[no] snapshot import <i>policy-name</i> Example: apic1(config)# snapshot import myImportPolicy	Creates a policy for importing snapshots.
Step 3	file <i>filename</i> Example: apic1(config-import)# file ce2_DailyAutoBackup-2015-11-21T01-00-17.tar.gz	Specifies the name of the file to be imported.
Step 4	action {merge replace} Example: apic1(config-import)# action replace	Specifies whether the imported configuration settings will be merged with the current settings or whether the imported configuration will completely replace the current configuration.
Step 5	[no] mode {atomic best-effort} Example: apic1(config-import)# mode atomic	Specifies how the import process handles configuration errors when applying the imported settings. The best-effort import mode allows skipping individual configuration errors in the archive, while atomic mode cancels the import upon any configuration error.

	Command or Action	Purpose
Step 6	(Optional) [no] remote path <i>remote-path-name</i> Example: <code>apic1(config-import)# remote path myBackupServer</code>	Specifies the name of a configured remote path from which the file will be imported. If no remote path is specified, the file is imported locally from a folder in the controller. The default is no remote path.
Step 7	end Example: <code>apic1(config-import)# end</code>	Returns to EXEC mode.
Step 8	Required: trigger snapshot import <i>policy-name</i> Example: <code>apic1# trigger snapshot import myImportPolicy</code>	Executes the snapshot import task.

Examples

This example shows how to configure and execute the importing of a snapshot file to replace the current configuration.

```

apic1# show snapshot files
File      : ce2_DailyAutoBackup-2015-11-21T01-00-17.tar.gz
Created   : 2015-11-21T01:00:21.167+00:00
Root      :
Size      : 22926

apic1# configure
apic1(config)# snapshot import myImportPolicy
apic1(config-import)# file ce2_DailyAutoBackup-2015-11-21T01-00-17.tar.gz
apic1(config-import)# action replace
apic1(config-import)# mode atomic
apic1(config-import)# end
apic1# trigger snapshot import myImportPolicy

```

Rollback Configuration Using Snapshots

The rollback feature provides an "undo" function that reverts changes made between one snapshot archive and a later snapshot archive. Only locally stored snapshot files are supported for rollback. You can optionally enable the preview mode to generate and view a rollback before implementing it.

SUMMARY STEPS

1. **configure**
2. **[no] snapshot rollback** *policy-name*
3. **first-file** *filename*
4. **second-file** *filename*
5. **[no] preview**
6. **end**

7. trigger snapshot rollback *policy-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] snapshot rollback <i>policy-name</i> Example: <code>apic1(config)# snapshot rollback myRollbackPolicy</code>	Creates a policy for rollback using snapshots.
Step 3	first-file <i>filename</i> Example: <code>apic1(config-rollback)# first-file ce2_DailyAutoBackup-2015-11-21T01-00-17.tar.gz</code>	Specifies the name of the earlier file.
Step 4	second-file <i>filename</i> Example: <code>apic1(config-rollback)# second-file ce2_DailyAutoBackup-2015-11-21T09-00-21.tar.gz</code>	Specifies the name of the later file.
Step 5	[no] preview Example: <code>apic1(config-rollback)# preview</code>	(Optional) Specifies that the rollback changes are generated and previewed but not applied. When preview mode is enabled, no changes to the configuration are made. After previewing rollback changes, use the no preview command to exit preview mode and enable the rollback to be applied when you reenter the trigger snapshot rollback commands.
Step 6	end Example: <code>apic1(config-rollback)# end</code>	Returns to EXEC mode.
Step 7	Required: trigger snapshot rollback <i>policy-name</i> Example: <code>apic1# trigger snapshot rollback myRollbackPolicy</code>	Executes the snapshot rollback task.

Examples

This example shows how to configure and execute a rollback without previewing it first.

```
apic1# show snapshot files
File      : ce2_DailyAutoBackup-2015-11-21T01-00-17.tar.gz
Created   : 2015-11-21T01:00:21.167+00:00
Root      :
Size      : 22926
```

```

File      : ce2_DailyAutoBackup-2015-11-21T09-00-21.tar.gz
Created   : 2015-11-21T09:00:24.025+00:00
Root      :
Size      : 23588

apicl# configure
apicl(config)# snapshot rollback myRollbackPolicy
apicl(config-rollback)# first-file ce2_DailyAutoBackup-2015-11-21T01-00-17.tar.gz
apicl(config-rollback)# second-file ce2_DailyAutoBackup-2015-11-21T09-00-21.tar.gz
apicl(config-rollback)# end
apicl# trigger snapshot rollback myRollbackPolicy

```

Uploading or Downloading a Snapshot File to a Remote Path

You can upload snapshot archive files from local storage to a remote path. You can also download snapshot archive files from the remote path.

Before you begin

You must configure a remote path to receive the file. See [Configuring a Remote Path for File Export, on page 505](#).

SUMMARY STEPS

1. **configure**
2. **[no] snapshot {upload | download} policy-name remote-path-name**
3. **remote path remote-path-name**
4. **file filename**
5. **end**
6. **trigger snapshot {upload | download} policy-name**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	[no] snapshot {upload download} policy-name remote-path-name Example: apicl(config)# snapshot upload myUpPolicy	Creates a policy for uploading or downloading snapshot files with a remote path.
Step 3	remote path remote-path-name Example: apicl(config-upload)# remote path myBackupServer	Specifies the name of a configured remote path to which the snapshot file will be sent.

	Command or Action	Purpose
Step 4	file <i>filename</i> Example: <pre>apic1(config-upload)# file ce2_DailyAutoBackup-2015-11-21T01-00-17.tar.gz</pre>	Specifies the name of the snapshot file to be sent.
Step 5	end Example: <pre>apic1(config-upload)# end</pre>	Returns to EXEC mode.
Step 6	trigger snapshot {upload download} policy-name Example: <pre>apic1# trigger snapshot upload myUpPolicy</pre>	Executes the snapshot upload or download task.

Examples

This example shows how to configure and execute the uploading of a snapshot file to a remote path.

```
apic1# show snapshot files
File      : ce2_DailyAutoBackup-2015-11-21T01-00-17.tar.gz
Created   : 2015-11-21T01:00:21.167+00:00
Root      :
Size      : 22926

apic1# configure
apic1(config)# snapshot upload myUpPolicy
apic1(config-upload)# remote path myBackupServer
apic1(config-upload)# file ce2_DailyAutoBackup-2015-11-21T01-00-17.tar.gz
apic1(config-upload)# end
apic1# trigger snapshot upload myUpPolicy
```

Managing Snapshot Files and Jobs

The following commands are available for managing snapshot files and jobs.

Command	Description
clear snapshot file <i>filename</i>	Removes a snapshot file from the local storage.
clear snapshot job <i>job-name</i>	Removes a snapshot job from the history.
show snapshot files	Displays the snapshot files in local storage.
show snapshot jobs	Displays recent snapshot tasks.
show snapshot active jobs	Displays currently-active snapshot tasks.

Examples

This example shows how to display snapshot files and the snapshot job history.

```
apic1# show snapshot files
File      : ce2_DailyAutoBackup-2015-11-21T01-00-17.tar.gz
Created   : 2015-11-21T01:00:21.167+00:00
Root      :
Size      : 22926

File      : ce2_DailyAutoBackup-2015-11-21T09-00-21.tar.gz
Created   : 2015-11-21T09:00:24.025+00:00
Root      :
Size      : 23588

apic1# show snapshot jobs
Type      : export
Run       : 2015-11-21T01-00-17
State     : success
Details   : Success
File Name : ce2_DailyAutoBackup-2015-11-21T01-00-17.tar.gz

Type      : export
Run       : 2015-11-21T09-00-21
State     : success
Details   : Success
File Name : ce2_DailyAutoBackup-2015-11-21T09-00-21.tar.gz

Type      : rollback
Run       : 2015-11-22T00-25-06
State     : running
Details   :
File Name : not applicable

apic1# clear snapshot job 2015-11-22T00-25-06
```



CHAPTER 19

Configuring Monitoring

- [Configuring Syslog, on page 491](#)
- [Configuring Call Home, on page 494](#)
- [Configuring TACACS External Logging, on page 502](#)
- [Sending an On-Demand Techsupport File Using the NX-OS Style CLI, on page 504](#)
- [Configuring a Remote Path for File Export, on page 505](#)
- [Using Show Commands for Monitoring, on page 507](#)
- [Configuring SNMP, on page 513](#)
- [Configuring SNMP Policy Using CLI, on page 514](#)
- [Configuring Smart Callhome, on page 516](#)

Configuring Syslog

Configuring a Logging Server Group

In the ACI fabric, one or more logging server-groups can be configured with one or more logging destination servers.

SUMMARY STEPS

1. **configure**
2. **logging server-group** *server-group-name*
3. **[no] description** *text*
4. **[no] console** [**severity** {**alerts** | **critical** | **emergencies**}]
5. **[no] logfile** [**severity** {**alerts** | **critical** | **debugging** | **emergencies** | **errors** | **information** | **notifications** | **warnings**}]
6. **[no] server** *ip-address-or-hostname* [**facility** *local-level*] [**severity** *severity-level*] [**mgmtepg** {**inb** | **oob**}] [**port** *port-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	logging server-group <i>server-group-name</i> Example: apicl(config)# logging server-group myLoggingGroup	Configure a grouping of servers for monitoring.
Step 3	[no] description <i>text</i> Example: apicl(config-logging)# logging description "This is my logging server group"	Specifies
Step 4	[no] console [severity { alerts critical emergencies }] Example: apicl(config-logging)# console severity critical	Enables logging to the console (only for switches) and optionally sets the minimum severity level for logging.
Step 5	[no] logfile [severity { alerts critical debugging emergencies errors information notifications warnings }] Example: apicl(config-logging)# logfile severity critical	Enables logging to the logfile and optionally sets the minimum severity level for logging.
Step 6	[no] server <i>ip-address-or-hostname</i> [facility <i>local-level</i>] [severity <i>severity-level</i>] [mgmtepg { inb oob }] [port <i>port-number</i>] Example: apicl(config-logging)# server reach.example.com level local4 mgmtepg inb port 514	Adds a destination logging server and optionally sets the minimum severity level for logging. • facility—Local facility in the form local<i>n</i> • severity—Minimum severity level for logging. Can be one of the options shown in the logfile command. • mgmt—Management endpoint group, either inb (inband) or oob (out of band). • port—Service port number of the logging server.

Examples

This example shows how to configure a syslog destination server group.

```

apicl# configure
apicl(config)# logging server-group myLoggingGroup
apicl(config-logging)# logging description "This is my logging server group"
apicl(config-logging)# console severity critical
apicl(config-logging)# logfile severity critical
apicl(config-logging)# server reach.example.com level local4 mgmtepg inb port 514

```


What to do next

Configure syslog with this logging server group as the logging destination.

Configuring Syslog

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.

Before you begin

Configure a logging server-group containing the servers to which syslog messages will be sent.

SUMMARY STEPS

1. **configure**
2. **syslog common**
3. **[no] logging description *text***
4. **[no] logging severity {alerts | critical | debugging | emergencies | errors | information | notifications | warnings}**
5. **[no] logging server-group *server-group-name***
6. **[no] logging audit**
7. **[no] logging event**
8. **[no] logging fault**
9. **[no] logging session**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	syslog common Example: <code>apic1(config)# syslog common</code>	Enters syslog common policy configuration mode.
Step 3	[no] logging description <i>text</i> Example: <code>apic1(config-syslog)# logging description "This is the common logging policy"</code>	Adds descriptive text about the policy.
Step 4	[no] logging severity {alerts critical debugging emergencies errors information notifications warnings} Example:	Specifies the minimum severity level for sending syslog messages.

	Command or Action	Purpose
	<code>apicl(config-syslog)# logging severity notifications</code>	
Step 5	[no] logging server-group <i>server-group-name</i> Example: <code>apicl(config-syslog)# logging server-group myLoggingGroup</code>	Specifies a destination logging server group.
Step 6	[no] logging audit Example: <code>apicl(config-syslog)# logging audit</code>	Enables audit logs to the policy.
Step 7	[no] logging event Example: <code>apicl(config-syslog)# logging event</code>	Enables event logs to the policy.
Step 8	[no] logging fault Example: <code>apicl(config-syslog)# logging fault</code>	Enables fault logs to the policy.
Step 9	[no] logging session Example: <code>apicl(config-syslog)# logging session</code>	Enables session logs to the policy.

Examples

This example shows how to configure syslog for messages of 'notification' severity or higher. Syslog messages from fault and event logs are sent to servers in server-group myLoggingGroup.

```

apicl# configure
apicl(config)# syslog common
apicl(config-syslog)# logging description "This is the common logging policy"
apicl(config-syslog)# logging severity notifications
apicl(config-syslog)# logging server-group myLoggingGroup
apicl(config-syslog)# logging audit
apicl(config-syslog)# logging event

```

Configuring Call Home

Configuring the Call Home Policy

In the ACI fabric, Cisco Call Home configuration can be added in the common monitoring policy.

SUMMARY STEPS

1. **configure**
2. **callhome common**
3. **[no] logging audit**
4. **[no] logging event**
5. **[no] logging fault**
6. **[no] logging severity {alert | critical | debug | emergency | error | info | notice | warning}**
7. **[no] periodic-inventory notification schedule *scheduler***
8. **show callhome common [destination-profile | query-profile | transport-email]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	callhome common Example: <code>apic1(config)# callhome common</code>	Enters Call Home common policy configuration mode.
Step 3	[no] logging audit Example: <code>apic1(config-callhome)# logging audit</code>	Enables audit logs to the policy.
Step 4	[no] logging event Example: <code>apic1(config-callhome)# logging event</code>	Enables event logs to the policy.
Step 5	[no] logging fault Example: <code>apic1(config-callhome)# logging fault</code>	Enables fault logs to the policy.
Step 6	[no] logging severity {alert critical debug emergency error info notice warning} Example: <code>apic1(config-callhome)# logging severity notice</code>	Specifies the minimum severity level for logging.
Step 7	[no] periodic-inventory notification schedule <i>scheduler</i> Example: <code>apic1(config-callhome)# periodic-inventory notification schedule EveryEightHours</code>	Configures a periodic notification scheduler. The scheduler must be previously configured.
Step 8	show callhome common [destination-profile query-profile transport-email] Example:	Shows Call Home configuration.

	Command or Action	Purpose
	apicl(config-callhome)# show callhome common	

Examples

This example shows how to configure a basic Call Home policy.

```

apicl# configure
apicl(config)# callhome common
apicl(config-callhome)# logging event
apicl(config-callhome)# logging fault
apicl(config-callhome)# logging severity notice
apicl(config-callhome)# periodic-inventory notification schedule EveryEightHours
apicl(config-callhome)# end
apicl# show callhome common
Callhome : common

```

```

Logging Enabled   : event,faults
Logging Severity : notice

```

Destination-Profile :

```

Admin State       : Enabled
Contract-id      : 12345678
Customer-id      : ABCDEFG
Email-addr       : admin@example.com
From email-addr  : apicl@example.com
Reply-To email-addr : apicl@example.com
Phone Number     : +14085551212
SMTP Port num    : 25

```

SMTP Server : smtp.example.com

Destination	Email-addr	Format	Message-Size	Message-Level
SanJose	admin@example.com	xml	40000	alert

Query-Profile :

Query Name	Query Type	Dn/Class	Target	Responses Subtree	Response Include
myUserQuery	class	User	self	children	
ep-records,	fault-records,	stats			

What to do next

Configure a destination profile and (optionally) a query profile.

Configuring a Call Home Destination Profile

You must configure at least one destination profile for Call Home. If the destination profile uses email message delivery, you must specify a Simple Mail Transfer Protocol (SMTP) server.

SUMMARY STEPS

1. **configure**
2. **callhome common**
3. **[no] destination-profile**
4. **[no] destination** *dest-name*
5. **[no] email-addr** *email*
6. **[no] format** {*aml* | *xml* | *short-txt*}
7. **[no] message-level** {*alert* | *critical* | *debug* | *emergency* | *error* | *info* | *notice* | *warning*}
8. **[no] message-size** *size*
9. **exit**
10. Configure the destination profile.
11. **show callhome common** [*destination-profile* | *query-profile* | *transport-email*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	callhome common Example: <code>apic1(config)# callhome common</code>	Enters Call Home common policy configuration mode.
Step 3	[no] destination-profile Example: <code>apic1(config-callhome)# destination-profile</code>	Configures a destination profile.
Step 4	[no] destination <i>dest-name</i> Example: <code>apic1(config-callhome-destnprof)# destination SanJose</code>	Configures a destination where the Call Home messages will be sent, including the format of the messages and the severity level for sending. Note You can configure more than one destination.
Step 5	[no] email-addr <i>email</i> Example: <code>apic1(config-callhome-destnprof-destn)# email-addr alerts@example.com</code>	Configures the e-mail address that will receive the Call Home messages. Up to 255 alphanumeric characters are accepted in e-mail address format.
Step 6	[no] format { <i>aml</i> <i>xml</i> <i>short-txt</i> } Example: <code>apic1(config-callhome-destnprof-destn)# format xml</code>	Configures the format for Call Home messages, which can be sent in the following formats: • aml—Adaptive Messaging Language (AML) XML schema definition (XSD) • xml—The XML format enables communication with the Cisco Systems Technical Assistance Center (TAC).

	Command or Action	Purpose
		• short-txt—Short text format provides a one or two line description of the fault that is suitable for pagers or printed reports.
Step 7	[no] message-level {alert critical debug emergency error info notice warning} Example: <pre>apicl(config-callhome-destnprof-destn)# message-level alert</pre>	Configures the minimum severity level for sending messages.
Step 8	[no] message-size size Example: <pre>apicl(config-callhome-destnprof-destn)# message-size 40000</pre>	Configures the size of the messages. The range is 0 to 5000000 characters.
Step 9	exit Example: <pre>apicl(config-callhome-destnprof-destn)# exit</pre>	Returns to destination profile configuration mode.
Step 10	Configure the destination profile. Example: <pre>apicl(config-callhome-destnprof)# (various commands)</pre>	Use the commands in Call Home Destination Profile Configuration Commands , on page 499
Step 11	show callhome common [destination-profile query-profile transport-email] Example: <pre>apicl(config-callhome-destnprof)# show callhome common transport-email</pre>	Shows Call Home configuration.

Examples

This example shows how to configure Call Home to send email messages of severity 'alert' or higher to alerts@example.com.

```
apicl# configure
apicl(config)# callhome common
apicl(config-callhome)# destination-profile
apicl(config-callhome-destnprof)# destination SanJose
apicl(config-callhome-destnprof-destn)# email-addr alerts@example.com
apicl(config-callhome-destnprof-destn)# format xml
apicl(config-callhome-destnprof-destn)# message-level alert
apicl(config-callhome-destnprof-destn)# message-size 40000
apicl(config-callhome-destnprof-destn)# exit
apicl(config-callhome-destnprof)# contract-id 12345678
apicl(config-callhome-destnprof)# customer-id ABCDEFG
apicl(config-callhome-destnprof)# description "Example Corporation"
apicl(config-callhome-destnprof)# site-id XYZ123
apicl(config-callhome-destnprof)# street-address "1 Cisco Way"
```

```

apic1(config-callhome-destnprof)# phone-contact +14085551212
apic1(config-callhome-destnprof)# email-contact admin@example.com
apic1(config-callhome-destnprof)# transport email from apic1@example.com
apic1(config-callhome-destnprof)# transport email reply-to apic1@example.com
apic1(config-callhome-destnprof)# transport email mail-server smtp.example.com mgmtepg inb
port 25
apic1(config-callhome)# end
apic1# show callhome common transport-email
From email-addr : apic1@example.com
SMTP Port num   : 25

SMTP Server : smtp.example.com

```

Call Home Destination Profile Configuration Commands

These commands are entered in the Call Home destination profile (**config-callhome-destnprof**) configuration mode.

Command	Purpose
contract-id <i>contract-id</i>	The Call Home contract number for the customer.
customer-id <i>customer-id</i>	The CCO ID that includes the contract numbers for the support contract in its entitlements.
description <i>text</i>	Descriptive text about this customer site.
email-contact <i>email</i>	The email address for the main contact.
phone-contact <i>phone-num</i>	The telephone number for the main contact.
site-id <i>site-id</i>	The unique Call Home identification number for the customer site.
street-address <i>address</i>	The mailing address for the main contact.
transport email from <i>email</i>	The email address that should appear in the From field on Call Home alert messages sent by the system.
transport email reply-to <i>email</i>	The return email address that should appear in the From field on Call Home alert messages sent by the system.
transport email mail-server <i>smtp-server</i> mgmtepg { <i>inb</i> <i>oob</i> } port <i>port-number</i>	The IP address or hostname of the SMTP server and the port number the system should use to talk to the SMTP server.

Configuring a Call Home Query

When an event triggers the sending of a Call Home report, information from your selected queries is included in the report. You can configure a query based on a class name or a distinguished name, and you can further qualify the query based on subtrees.

Before you begin

SUMMARY STEPS

1. **configure**
2. **callhome common**
3. **[no] query-profile**
4. **[no] query *query-name* type {class *class-name* | dn *name*}**
5. **[no] response-subtree {full | children | no}**
6. **[no] response-incl {*option*[,*option*[,*option*...]]}**
7. **[no] target {children | self | subtree}**
8. **show callhome common [destination-profile | query-profile | transport-email]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	callhome common Example: apicl(config)# callhome common	Enters Call Home common policy configuration mode.
Step 3	[no] query-profile Example: apicl(config-callhome)# query-profile	Enters Call Home query profile configuration mode.
Step 4	[no] query <i>query-name</i> type {class <i>class-name</i> dn <i>name</i>} Example: apicl(config-callhome-queryprof)# query myUserQuery type class User	Configures a query profile.
Step 5	[no] response-subtree {full children no} Example: apicl(config-callhome-queryprof-query)# response-subtree children	Configures the response subtree. You can choose to include the full subtree, only children, or no subtree information.
Step 6	[no] response-incl {<i>option</i>[,<i>option</i>[,<i>option</i>...]]} Example: apicl(config-callhome-queryprof-query)# response-incl ep-records,fault-records,stats	Configures the specific subtree information categories to be included in the response. Multiple categories can be specified in a comma-separated list. The available categories are listed in Query Subtree Categories, on page 501 .
Step 7	[no] target {children self subtree} Example: apicl(config-callhome-queryprof-query)# target self	Configures the query target.

	Command or Action	Purpose
Step 8	show callhome common [destination-profile query-profile transport-email] Example: <pre>apic1(config-callhome-queryprof-query)# show callhome common query-profile</pre>	Shows Call Home configuration.

Examples

This example shows how to configure a Call Home query.

```
apic1# configure
apic1(config)# callhome common
apic1(config-callhome)# query-profile
apic1(config-callhome-queryprof)# query myUserQuery type class User
apic1(config-callhome-queryprof-query)# response-subtree children
apic1(config-callhome-queryprof-query)# response-incl ep-records,fault-records,stats
apic1(config-callhome-queryprof-query)# target self
apic1(config-callhome)# end
apic1# show callhome common destination-profile
Query-Profile :
```

Query Name	Query Type	Dn/Class	Target	Responses Subtree	Response Include
myUserQuery	class	User	self	children	
ep-records,fault-records,stats					

Query Subtree Categories

Query Category	Description
add-mo-list	
audit-logs	
config-only	
count	
custom-path-hop	
deployment	
deployment-records	
ep-records	
event-logs	
fault-count	

fault-records	
faults	
full-deployment	
health	
health-records	
local-prefix	
no-scoped	
none	
port-deployment	
record-subtree	
relations	
relations-with-parent	
required	
state	
stats	
tasks	

Configuring TACACS External Logging

Creating a TACACS External Logging Destination Group Using the NX-OS-Style CLI

You can use the NX-OS-style command line interface (CLI) to configure TACACS destination groups and destinations. A TACACS destination group enables you to create a list of remote TACACS server destinations to which AAA logging data is sent. You can create one or more destinations in a group. After the destination group is created, you can associate it with a TACACS source, either for a fabric policy, an external access policy, or a specific tenant policy configured on the Cisco Application Policy Infrastructure Controller (Cisco APIC).



Note

You must have administrator rights to access the TACACS External Logging commands in the NX-OS-style CLI.

The following example CLI commands show how to configure a TACACS destination group and destination using the NX-OS-style CLI:

Step 1 Enter the configuration mode.

Example:

```
apicl# config
```

Step 2 Create a TACACS destination group.

Example:

In the following command, a TACACS destination group named "tacacs-dest-grp-1" is created:

```
apicl(config)# tacacslog-group tacacs-dest-grp-1
```

Step 3 Create a TACACS destination in the new destination group.

Example:

In the following command, a remote TACACS destination with an IP address of "1.1.1.1" is created and includes the default port number 49:

```
apicl(config-tacacslog-group)# remote-dest 1.1.1.1 port 49
```

Note You can have logs sent to multiple ports on the same IP address by including additional port numbers after the port keyword.

Step 4 Configure specific parameters for the new remote TACACS destination.

Example:

In the following command example, the following characteristics are configured for the new remote destination:

- Authentication key: 12345
- Authentication protocol: MS-CHAP
- Management EPG: Out-of-Band

```
apicl(config-remote-dest)# key
Enter Key: 12345
Enter Key again: 12345
apicl(config-remote-dest)# protocol mschap
apicl(config-remote-dest)# management-epg oob
```

The result of this configuration is the creation of a TACACS destination group containing a remote TACACS server destination. If you want the same AAA logging data sent to multiple remote TACACS servers, you can repeat steps 3 and 4 as many times as needed.

Creating a TACACS External Logging Source Using the NX-OS-Style CLI

You can use the NX-OS-style CLI to configure TACACS sources. In this configuration, the source is associated with a TACACS destination group. Where a TACACS source is created determines which set of AAA logging data is sent. For example, if you create the TACACS source in Fabric Policies, all AAA logging data for the Cisco Application Centric Infrastructure (Cisco ACI) fabric supported by Cisco Application Policy Infrastructure

Controller (Cisco APIC) is sent to the associated TACACS destinations. You can create one or more sources to support different destination groups.

The following example CLI commands show how to configure a TACACS source using the NX-OS-style CLI:

Step 1 Enter the configuration mode.

Example:

```
apic1# config
```

Step 2 Create a TACACS source.

Example:

In the following command, a TACACS source named "tacacs-src-1" is created:

```
apic1(config)# tacacslog-monitoring common tacacslog-src tacacs-src-1
```

Step 3 Associate the TACACS source with a TACACS destination group.

Example:

In the following command, a TACACS destination group named "tacacs-dest-grp-1" is associated with the new TACACS source:

```
apic1(config-tacacslog-monitoring)# server-group tacacs-dest-grp-1
```

The result of this configuration is the creation of a TACACS source for the entire fabric and the association of a destination group containing a remote TACACS server destination. All AAA logging data for the entire fabric is then sent to the associated TACACS destination(s).

Sending an On-Demand Techsupport File Using the NX-OS Style CLI



Note

Do not trigger techsupport file collection from more than five nodes simultaneously, especially if they are to be exported into the APIC or to an external server with insufficient bandwidth and compute resources.

To avoid excessive storage usage in APIC, remove locally-stored techsupport files promptly.

Before you begin

Configure a remote path for exporting the techsupport file.

SUMMARY STEPS

1. **trigger techsupport** {all | controllers switch *node-id*} [**remotename** *remote-path-name*]
2. **trigger techsupport host** *host-id*
3. **trigger techsupport local**

4. show techsupport {all | controllers switch *node-id*} status

DETAILED STEPS

	Command or Action	Purpose
Step 1	trigger techsupport {all controllers switch <i>node-id</i>} [remotename <i>remote-path-name</i>] Example: <pre>apic1# trigger techsupport switch 101,103 remotename remote5</pre>	Triggers the export of a techsupport file from the controllers, switches, or all to the remote path. For switches, you can specify a range or a comma-separated list. If no remote host is specified, the file is collected in the controller itself.
Step 2	trigger techsupport host <i>host-id</i> Example: <pre>apic1# trigger techsupport host</pre>	Triggers the export of a techsupport file from the specified host to the remote host. If no remote host is specified, the file is collected in the controller itself.
Step 3	trigger techsupport local Example: <pre>apic1# trigger techsupport local</pre>	Triggers the export of a local techsupport file to the remote host. If no remote host is specified, the file is collected in the controller itself.
Step 4	show techsupport {all controllers switch <i>node-id</i>} status Example: <pre>apic1# show techsupport switch 101 status</pre>	After a techsupport file is triggered, this command shows the status of the techsupport report.

Examples

This example shows how to trigger a techsupport file for switch 101, to be stored locally on the apic1 controller.

```
apic1# trigger techsupport switch 101
```

Triggering techsupport for Switch 101 using policy supNode101, setting filters to default value

Triggered on demand tech support successfully for Switch 101, will be available at:
/data/techsupport on
the controller. Use 'show techsupport' with your options to check techsupport status.

Configuring a Remote Path for File Export

In the ACI fabric, you can configure one or more remote destinations for exporting techsupport or configuration files.

SUMMARY STEPS

1. configure
2. [no] remote path *remote-path-name*

3. **user** *username*
4. **path** {ftp | scp | sftp} *host[:port]* [**remote-directory**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apicl# configure	Enters global configuration mode.
Step 2	[no] remote path <i>remote-path-name</i> Example: apicl(config)# remote path myFiles	Enters configuration mode for a remote path.
Step 3	user <i>username</i> Example: apicl(config-remote)# user admin5	Sets the user name for logging in to the remote server. You are prompted for a password.
Step 4	path {ftp scp sftp} <i>host[:port]</i> [remote-directory] Example: apicl(config-remote)# path sftp filehost.example.com:21 remote-directory /reports/apic	Sets the path and protocol to the remote server. You are prompted for a password.

Examples

This example shows how to configure a remote path for exporting files.

```

apicl# configure
apicl(config)# remote path myFiles
apicl(config-remote)# user admin5
You must reset the password when modifying the path:
Password:
Retype password:
apicl(config-remote)# path sftp filehost.example.com:21 remote-directory /reports/apic
You must reset the password when modifying the path:
Password:
Retype password:

```

Using Show Commands for Monitoring

About Using the Show Commands

The show commands for faults, events, health, statistics, and audit logs can be filtered to display specific types of information or information from specific entities, such as controllers, leaf switches, spine switches, or tenants.

Broad queries are expensive in terms of system resources and storage. For example, using the show faults, events, or audit commands without entity filters retrieves all logs or records from the entire system. We recommend that you make use of the available data and entity filters to narrow your query as much as possible. For example, the following command would result in a quicker and more filtered response by limiting the query to the most recent 45 minute period:

```
show audits last-minutes 45
```



Tip At each point in the command, typing ‘?’ displays all possible keywords and options that can be used at that point along with a brief explanation of each.

Using the show faults Command

The **show faults** command can combine several data filters and an entity filter to deliver a specific set of faults. The command syntax is:

```
show faults [filter1 [filter2... ]] [entity-filter]
```

Entity filters restrict the output to faults of a controller, leaf, spine, or tenant. The available entity filters are listed in [Entity Filters for Show Commands, on page 512](#).

Data filters are provided to make the task of querying faults easier for the user. The available data filters are:

Filter	Description
ack {yes no}	acknowledgment status
cause <i>name</i>	cause
code <i>fault-code</i>	fault code
controller	controller information
detail	detailed faults information
end-time <i>YYYY-MM-DDTHR-MM:SS</i>	fault activity up to this time
history	historical information
id <i>fault-id</i>	fault ID
l4l7-cluster [<i>cluster name</i> <i>tenant name</i>]	L4 L7 device information

Filter	Description
l4l7-graph [<i>cluster name</i> <i>tenant name</i>]	L4 L7 graph information
last-days <i>days</i>	fault activity in the last N days
last-hours <i>hours</i>	fault activity in the last N hours
last-minutes <i>minutes</i>	fault activity in the last N minutes
lc <i>lc-state</i>	lifecycle state
leaf [<i>leaf-id</i>]	leaf switch information
microsoft domain <i>name</i>	Microsoft domain information
min-severity <i>severity-value</i>	minimum severity
severity <i>severity-value</i>	severity
spine [<i>spine-id</i>]	spine switch information
start-time <i>YYYY-MM-DDTHR-MM:SS</i>	fault activity starting from this time
tenant [<i>name</i>]	tenant information
type <i>fault-type</i>	fault type
vmware domain <i>name</i>	VMware domain information

Examples

This example shows all faults that occurred in the past five days with code “F110473”, severity “warning”, lifecycle “raised” and acknowledgment status “no” for the tenant TSW_Tenant0.

```
apic1# show faults code F110473 last-days 5 severity warning lc raised ack no tenant
TSW_Tenant0
Code          : F110473
Severity      : warning
Last Transition : 2015-11-03T01:19:04.913+00:00
Lifecycle     : raised
DN           : uni/tn-TSW_Tenant0/BD-tsw0ctx0BD1/fault-F110473
Description   : TCA: ingress drop bytes rate(l2IngrBytesAg15min:dropRate)
                value 160462 raised above threshold 100000
```

Using the show events Command

The **show events** command can combine several data filters and an entity filter to deliver a specific set of events. The command syntax is:

```
show events [filter1 [filter2... ]] [entity-filter]
```

Entity filters restrict the output to events of a controller, leaf, spine, or tenant. The available entity filters are listed in [Entity Filters for Show Commands, on page 512](#).

Data filters are provided to make the task of querying events easier for the user. The available data filters are:

Filter	Description
cause <i>fault-value</i>	cause
code <i>event-code</i>	event code
controller	controller information
detail	detailed events information
end-time <i>YYYY-MM-DDTHR-MM:SS</i>	event activity up to this time
id <i>event-id</i>	event ID
last-days <i>days</i>	event activity in the last N days
last-hours <i>hours</i>	event activity in the last N hours
last-minutes <i>minutes</i>	event activity in the last N minutes
leaf [<i>leaf-id</i>]	leaf switch information
spine [<i>spine-id</i>]	spine switch information
start-time <i>YYYY-MM-DDTHR-MM:SS</i>	event activity starting from this time
tenant [<i>name</i>]	tenant information

Examples

This example shows all events on leaf 101.

```
apic1# show events leaf 101

Severity          : info
Affected Object   : topology/pod-1/node-101/sys/phys-[eth1/28]
Code              : E4208843
ID                : 8589934758
Cause             : transition
Description       : PhysIf eth1/28 modified
Creation Time     : 2015-11-03T01:11:16.763+00:00
```

Using the show health Command

The **show health** command can combine several data filters and an entity filter to deliver a specific health report. The command syntax is:

```
show health [filter1 [filter2... ]] [entity-filter]
```

Entity filters restrict the output to health scores of a controller, leaf, spine, or tenant. The available entity filters are listed in [Entity Filters for Show Commands, on page 512](#).

Data filters are provided to make the task of querying health easier for the user. The available data filters are:

Filter	Description
end-time <i>YYYY-MM-DDTHR-MM:SS</i>	health activity up to this time
history	historical information
max-change <i>percentage</i>	minimum change in health score percentage
min-hs <i>score</i>	maximum health score
start-time <i>YYYY-MM-DDTHR-MM:SS</i>	health activity starting from this time

Examples

This example shows a brief health report for all tenants.

```
apic1# show health tenant
Tenant                Score    Change (%)    Created
-----
infra                  100        0            2015-05-12 18:45:47PDT
common                 100        0            2015-05-12 18:45:47PDT
TSW_Tenant0            98         0            2015-05-12 18:20:58PDT
mgmt                   100        0            2015-05-12 18:45:47PDT
```

This example shows all historical health records from the 4th of November that have a maximum health score of 75 that have had a minimum change of 10% for the tenant TSW_Tenant0.

```
apic1# show health max-hs 75 min-change 10 start-time 2015-11-04T01:55:48 history tenant
TSW_Tenant0
```

Using the show audits Command

The **show audits** command can be used to view the audit-logs as well as the session logs for an entity. The command can combine several data filters and an entity filter to deliver a specific set of audit logs. The command syntax is:

show audits [*filter1* [*filter2...*]] [*entity-filter*]

Entity filters restrict the output to logs of a controller, leaf, spine, or tenant. The available entity filters are listed in [Entity Filters for Show Commands, on page 512](#).

Data filters are provided to make the task of querying audit logs easier for the user. The available data filters are:

Filter	Description
action { creation deletion failure modification special state-transition }	object action indicator
controller	controller information
detail	detailed log information

Filter	Description
end-time <i>YYYY-MM-DDTHR-MM:SS</i>	log activity up to this time
id <i>log-id</i>	log ID
last-days <i>days</i>	log activity in the last N days
last-hours <i>hours</i>	log activity in the last N hours
last-minutes <i>minutes</i>	log activity in the last N minutes
leaf [<i>leaf-id</i>]	leaf switch information
spine [<i>spine-id</i>]	spine switch information
start-time <i>YYYY-MM-DDTHR-MM:SS</i>	log activity starting from this time
tenant [<i>name</i>]	tenant information
user <i>user-name</i>	name of user

Examples

This example shows all audit logs in the last 45 minutes for the tenant TSW_Tenant0.

```
apic1# show audits last-minutes 45 tenant TSW_Tenant0
Creation Time   : 2015-11-03T01:11:05.708+00:00
ID             : 12884902085
User           : admin
Action         : creation
Affected Object : uni/tn-TSW_Tenant0/out-T0-sub-L3OUT-1/instP-
                l3extInstP-1/extsubnet-[192.5.1.0/24]
Description    : Subnet 192.5.1.0/24 created
```

Using the show stats Command

The **show stats** command can combine data filters and an entity filter to deliver a specific set of statistics. The command syntax is:

show stats granularity *granularity* [**cumulative**] [**history**] [*entity-filter*]

Entity filters restrict the output to statistics of a leaf, spine, or tenant. The available entity filters are listed in [Entity Filters for Show Commands, on page 512](#).

Data filters are provided to make the task of querying statistics easier for the user. The available data filters are:

Filter	Description
cumulative	cumulative statistics information
granularity { 5min 15min 1h 1d 1w 1mo 1qtr 1year }	the sampling interval size which can be 5 minutes, 15 minutes, 1 hour, 1 day, 1 week, 1 month, 1 quarter, or 1 year

Filter	Description
history	historical statistics information

Examples

This example shows 15 minute granularity statistics for the tenant TSW_Tenant0.

```
apic1# show stats granularity 15min tenant TSW_Tenant0
```

This example shows 15 minute granularity statistics for a specific port.

```
apic1# show stats granularity 15min leaf 101 interface ethernet 1/1
```

Entity Filters for Show Commands

Entity filters can extend many show commands to restrict the output to faults of a controller, leaf, spine, or tenant. The available entity filters are:

Filter
controller
leaf <i>node-id</i> [fex]
leaf <i>node-id</i> interface [ethernet <i>slot/port</i> I3instance [<i>instance-name</i>] mgmt [mgmt0] portchannel tunnel [<i>tunnel-name</i>]]
leaf <i>node-id</i> inventory { chassis [<i>number</i>] fans [<i>number</i>] module [<i>number</i>] powersupply [<i>number</i>] supervisor [<i>number</i>]}
leaf <i>node-id</i> protocol { arp bgp coop ipv4 ipv6 isis lldp ospf ospfv3 }
leaf <i>node-id</i> vpc {
leaf <i>node-id</i> vrf [<i>vrf-name</i>]
spine <i>node-id</i>
spine <i>node-id</i> interface [ethernet <i>slot/port</i> I3instance [<i>instance-name</i>] mgmt [mgmt0] tunnel [<i>tunnel-name</i>]]
spine <i>node-id</i> inventory { chassis [<i>number</i>] fabric [<i>number</i>] fans [<i>number</i>] module [<i>number</i>] powersupply [<i>number</i>] supervisor [<i>number</i>] system [<i>number</i>]}
spine <i>node-id</i> protocol { arp bgp coop ipv4 ipv6 isis lldp ospf ospfv3 }
spine <i>node-id</i> vrf [<i>vrf-name</i>]
tenant <i>tenant-name</i>
tenant <i>tenant-name</i> application [<i>app-name</i>] [epg]

Filter
tenant <i>tenant-name</i> bridge-domain [<i>bd-name</i>]
tenant <i>tenant-name</i> interface bridge-domain [<i>bd-name</i>]

Configuring SNMP

Before you begin

To allow SNMP communications, you must configure an out-of-band contract allowing SNMP traffic, which is normally on UDP:161.

SUMMARY STEPS

1. **configure**
2. **template snmp-fabric** *snmp-fabric-template-name*
3. **[no] snmp-server protocol enable**
4. **[no] snmp-server community** *community-name*
5. **snmp-server contact** *contact-name*
6. **snmp-server location** *location-name*
7. **exit**
8. **template pod-group** *pod-group-template-name*
9. **inherit snmp-fabric** *snmp-fabric-template-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	template snmp-fabric <i>snmp-fabric-template-name</i> Example: <code>apic1(config)# template snmp-fabric Poll</code>	Enters template snmp-fabric mode.
Step 3	[no] snmp-server protocol enable Example: <code>apic1(config-template-snmp-fabric)# snmp-server protocol enable</code>	Enables (or disables) SNMP protocol support.
Step 4	[no] snmp-server community <i>community-name</i> Example: <code>apic1(config-template-snmp-fabric)# snmp-server community mysecret</code>	The community is required for SNMPv2 only.

	Command or Action	Purpose
Step 5	snmp-server contact <i>contact-name</i> Example: <code>apicl(config-template-snmp-fabric) # snmp-server contact admin80</code>	.
Step 6	snmp-server location <i>location-name</i> Example: <code>apicl(config-template-snmp-fabric) # snmp-server location SanJose</code>	Sets the location for the SNMP server.
Step 7	exit Example: <code>apicl(config-template-snmp-fabric) # exit</code>	Returns to global configuration mode
Step 8	template pod-group <i>pod-group-template-name</i> Example: <code>apicl(config) # template pod-group allPods</code>	Configures a pod-group template (policy).
Step 9	inherit snmp-fabric <i>snmp-fabric-template-name</i> Example: <code>apicl(config-pod-group) # inherit snmp-fabric Poll</code>	Associates the pod-profile with the previously configured pod group.

Examples

The following example configures an out-of-band contract allowing SNMP traffic in the fabric.

```
apicl# configure
apicl(config) # template snmp-fabric Poll
apicl(config-template-snmp-fabric) # snmp-server protocol enable
apicl(config-template-snmp-fabric) # snmp-server community mysecret
apicl(config-template-snmp-fabric) # snmp-server contact admin80
apicl(config-template-snmp-fabric) # snmp-server location SanJose
apicl(config-template-snmp-fabric) # exit
apicl(config) # template pod-group allPods
apicl(config-pod-group) # inherit snmp-fabric Poll
apicl(config-pod-group) # exit
apicl(config) #
```

Configuring SNMP Policy Using CLI

Use this procedure to configure SNMP policy.

SUMMARY STEPS

1. **configure**
2. **template snmp-fabric default**

3. **snmp-server clientgroup**
4. **snmp-server community**
5. **snmp-server contact**
6. **snmp-server host**
7. **snmp-server location**
8. **snmp-server protocol**
9. **snmp-server trap-fwd-server**
10. **snmp-server user**
11. **show running-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <pre>apic1# configure</pre>	Enters configuration mode.
Step 2	template snmp-fabric default Example: <pre>apic1(config)# template snmp-fabric default</pre>	Creates a SNMP policy.
Step 3	snmp-server clientgroup	Configures SNMP client group. A client group is a group of client IP addresses that allows SNMP access to routers or switches.
Step 4	snmp-server community Example: <pre>apic1(config-template-snmp-fabric)# snmp-server community abc</pre>	Configures SNMP community. The SNMP community profile enables access to the router or switch statistics for monitoring.
Step 5	snmp-server contact	Configures SNMP contact information.
Step 6	snmp-server host Example: <pre>apic1(config-template-snmp-fabric)# snmp-server host 2001:420:28e:2020::10 traps-version 2c abc apic1(config-template-snmp-fabric)# snmp-server host 2001:420:28e:2020::2 traps-version 2c abc apic1(config-template-snmp-fabric)# snmp-server host 2001:420:28e:2020::11 traps-version 2c abc</pre>	Configures SNMP trap host.
Step 7	snmp-server location	Configures SNMP location.
Step 8	snmp-server protocol Example:	Configures SNMP protocol.

	Command or Action	Purpose
	<code>apic1(config-template-snmp-fabric)# snmp-server protocol enable</code>	
Step 9	snmp-server trap-fwd-server Example: <code>apic1(config-template-snmp-fabric)# snmp-server trap-fwd-server 172.31.128.199</code>	Configures SNMP trap forwarding server.
Step 10	snmp-server user Example: <code>apic1(config-template-snmp-fabric)# snmp-server user test_user auth hmac-md5-96 '' priv none privacy-passphrase ''</code>	Configures SNMP user. The SNMP user profile is used to associate users with SNMP policies for monitoring devices in a network.
Step 11	show running-config	Verifies the configuration.

Configuring 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 NX-OS-Style CLI

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).

You can use the NX-OS-style CLI to configure Smart Callhome destination groups and destinations. A Smart Callhome destination group enables you to create a list of email destinations to which fault data is sent. You

can create one or more destinations in a group. After the destination group is created, you can associate it with a Smart Callhome source, either for the entire switch fabric supported by the Cisco Application Policy Infrastructure Controller (Cisco APIC) or for a specific Tenant.

The following example CLI commands show how to configure a Smart Callhome destination group and destination using the NX-OS-style CLI:

SUMMARY STEPS

1. Enter the configuration mode.
2. Enter the Smart Callhome common policy configuration mode.
3. Create a Smart Callhome destination group.
4. Configure an SMTP server in the new destination group.
5. Configure profile parameters about the new Smart Callhome destination group.
6. Create a Smart Callhome destination in the new destination group.
7. Configure specific parameters for the new remote Smart Callhome destination.

DETAILED STEPS

Step 1 Enter the configuration mode.

Example:

```
apic1# config
```

Step 2 Enter the Smart Callhome common policy configuration mode.

Example:

```
apic1(config)# smartcallhome common
```

Note The default name for the common policy configuration mode is "common". It is the only name that can be created.

Step 3 Create a Smart Callhome destination group.

Example:

In the following command, a Smart Callhome destination group is created:

```
apic1(config-smartcallhome)# destination-profile
```

Step 4 Configure an SMTP server in the new destination group.

Example:

In the following command, an SMTP server with an IP address of "10.10.10.2" is added to the destination group:

```
apic1(config-callhome-destnprof)# transport email mail-server 10.10.10.2
```

Step 5 Configure profile parameters about the new Smart Callhome destination group.

Example:

The following commands provide additional information about the destination group:

- **contract-id:** The service contract ID of the customer.
- **customer-id:** The customer ID.

- **description:** A description for the Smart Callhome destination profile.
- **email-contact:** The customer contact e-mail address.
- **phone-contact:** The customer contact phone number.
- **site-id:** The ID of the site where the network is deployed.
- **street-address:** The street address of the site.

Step 6 Create a Smart Callhome destination in the new destination group.

Example:

In the following command, a remote Smart Callhome destination named "sch-dest-1" is created:

```
apic1(config-callhome-destnprof)# destination sch-dest-1
```

Step 7 Configure specific parameters for the new remote Smart Callhome destination.

Example:

In the following command example, the following characteristics are configured for the new remote destination:

- Email address: net-admin@site.com
- Message format: Short text
- RFC Compliant: True

```
apic1(config-callhome-destnprof-destn)# email-addr net-admin@site.com
apic1(config-callhome-destnprof-destn)# format short-txt
apic1(config-callhome-destnprof-destn)# rfc-compliant true
```

The result of this configuration is the creation of a Smart Callhome destination group containing a remote email destination. If you want the same Smart Callhome fault data sent to multiple email destinations, you can repeat steps 5 and 6 as many times as needed.



CHAPTER 20

Configuring SPAN

- [Configuring SPAN and ERSPAN, on page 519](#)

Configuring SPAN and ERSPAN

In the ACI Fabric, SPAN feature can be configured in three categories:

- Access – for monitoring traffic originating from access ports in leaf nodes
- Fabric – for monitoring traffic from fabric ports in leaf or spine nodes
- Tenant – for monitoring traffic from endpoint groups (EPGs) within a tenant

The following table shows the different configuration elements for each session.

Session Type	Sources	Filters	Destination
Access Local	Access Ports, Port-channels local to one leaf	EPG	Port local to same leaf as sources
Access ERSPAN	Access Ports, Port-channels, VPCs among one or more leaf nodes	EPG	EPG anywhere in the fabric
Fabric ERSPAN	Fabric ports in one or more leaf or spine nodes	BD or VRF	EPG anywhere in the fabric
Tenant ERSPAN	EPG anywhere in the fabric	-	EPG anywhere in the fabric

SPAN Guidelines and Restrictions

- Use SPAN only for troubleshooting. SPAN traffic competes with user traffic for switch resources. To minimize the load, configure SPAN to copy only the specific traffic that you want to analyze.
- Span sessions with an uSeg EPG as the source are not supported.
- You cannot specify an l3extLifP layer 3 subinterface as a SPAN source. You must use the entire port for monitoring traffic from external sources.
- In local SPAN for FEX interfaces—the FEX interfaces can only be used as SPAN sources, not SPAN destinations. On Generation 1 switches (Cisco Nexus 9000 Series switches without EX or FX on the

switch name), Tx SPAN does not work for any Layer 3 switched traffic. On Generation 2 switches (with EX or FX on the switch name), Tx SPAN does not work whether traffic is Layer 2 or Layer 3 switched. There are no limitations for Rx SPAN.

- Tenant and access SPAN use the encapsulated remote extension of SPAN (ERSPAN) type I, while fabric SPAN uses ERSPAN type II. For information regarding ERSPAN headers, refer to the IETF Internet Draft at this URL: <https://tools.ietf.org/html/draft-foschiano-erspan-00>.
- ERSPAN destination IPs must be learned in the fabric as an endpoint.
- SPAN supports IPv6 traffic but the destination IP for the ERSPAN cannot be an IPv6 address.
- The individual port member of a port-channel or a vPC cannot be configured as the source. Use the port-channel, vPC, or vPC component as the source in the SPAN session.
- A fault is not raised on the ERSPAN source group when the destination EPG is deleted or unavailable.
- SPAN filters are supported on -EX, -FX, and -FX2 leaf switches only.
- When an EPG filter is enabled within a SPAN session, ARP packets, which are sent out of the interface in the transit, or tx, direction, will not be spanned.
- SPAN filters are not supported in the following:
 - Fabric ports
 - Fabric and tenant SPAN sessions
 - Spine switches
- L4 port range filter entries will not be added if you attempt to add more L4 port ranges than are officially supported.
- A SPAN session will not come up if you attempt to associate more than the supported filter entries at the SPAN source group level or at the individual SPAN source level.
- Deleted filter entries will remain in TCAM if you add or delete more filters entries than are officially supported.
- See the *Verified Scalability Guide for Cisco ACI* document for SPAN-related limits, such as the maximum number of active SPAN sessions and SPAN filter limitations.
- For the SPAN-on-drop feature, the following guidelines and restrictions apply:
 - The SPAN-on-drop feature is supported on -EX, -FX, and FX2 leaf switches.
 - The SPAN-on-drop feature only captures packets with forwarding drops in the LUX block, which captures forwarding drop packets at the ingress. The SPAN-on-drop feature cannot capture the BMX (buffer) and RWX (egress) drops.
 - When using the troubleshooting CLI to create a SPAN session with SPAN-on-drop enabled and Cisco APIC as the destination, the session is disabled when 100 MB of data is captured.
 - On a modular chassis, the SPAN-on-drop feature will only work for the packets dropped on the line cards. Packets that are dropped on the fabric card will not be spanned.
 - SPAN-on-drop ACLs with other SPAN ACLs are not merged. If a SPAN-on-drop session is configured on an interface along with ACL-based SPAN, then any packets dropped on that interface will only be sent to the SPAN-on-drop session.

- You cannot configure SPAN on drop and SPAN ACL on the same session.
- When an access or fabric port-drop session and a global-drop session are configured, the access or fabric port-drop session takes the priority over the global-drop session.
- The number of filter entries supported in TCAM = $(M * S1 * 1 + N * S2 * 2) + (S3 * 2)$. This is applicable to rx SPAN or tx SPAN, separately. Currently, the maximum filter entries supported in tx or rx SPAN is 480 in each direction when following this formula (and assuming there are no other sources that are configured without filter-group association [means $S3 = 0$] and with 16 port-ranges included). When the number of filter entries exceed the maximum number allowed, a fault will be raised. Note that you can specify Layer 4 port ranges in the filter entry. However, sixteen Layer 4 ports are programmed into the hardware as a single filter entry.

**Note**

- M=The number of IPv4 filters
- S1=The number of sources with IPv4 filters
- N=The number of IPv6 filters
- S2=The number of sources with IPv6 filters
- S3=The number of sources with no filter group association

Configuring Local SPAN in Access Mode

This is the traditional SPAN configuration local to an Access leaf node. Traffic originating from one or more access ports or port-channels can be monitored and sent to a destination port local to the same leaf node.

SUMMARY STEPS

1. **configure**
2. **[no] monitor access session** *session-name*
3. **[no] description** *text*
4. **[no] destination interface ethernet** *slot/port leaf node-id*
5. **[no] source interface ethernet** {[*fex/*]*slot/port | port-range*} **leaf** *node-id*
6. **[no] direction** {**rx** | **tx** | **both**}
7. **[no] filter tenant** *tenant-name application application-name epg epg-name*
8. **exit**
9. **[no] source interface port-channel** *port-channel-name-list leaf node-id [fex fex-id]*
10. **[no] shutdown**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.

	Command or Action	Purpose
Step 2	[no] monitor access session <i>session-name</i> Example: <code>apic1(config)# monitor access session mySession</code>	Creates an access monitoring session configuration.
Step 3	[no] description <i>text</i> Example: <code>apic1(config-monitor-access)# description "This is my SPAN session"</code>	Adds a description for this access monitoring session. If the text includes spaces, it must be enclosed in single quotes.
Step 4	[no] destination interface ethernet <i>slot/port leaf node-id</i> Example: <code>apic1(config-monitor-access)# destination interface eth 1/2 leaf 101</code>	Specifies the destination interface. The destination interface cannot be a FEX port or port-channel.
Step 5	[no] source interface ethernet {[fex/]slot/port port-range} leaf <i>node-id</i> Example: <code>apic1(config-monitor-access)# source interface eth 1/2 leaf 101</code>	Specifies the source interface port or port range.
Step 6	[no] direction {rx tx both} Example: <code>apic1(config-monitor-access-source)# direction tx</code>	Specifies direction of traffic to be monitored. The direction can be configured independently for each source port range.
Step 7	[no] filter tenant <i>tenant-name application application-name</i> epg <i>epg-name</i> Example: <code>apic1(config-monitor-access-source)# filter tenant t1 application appl epg epg1</code>	Filters traffic to be monitored. The filter can be configured independently for each source port range.
Step 8	exit Example: <code>apic1(config-monitor-access-source)# exit</code>	Returns to access monitor session configuration mode.
Step 9	[no] source interface port-channel <i>port-channel-name-list leaf node-id</i> [fex <i>fex-id</i>] Example: <code>apic1(config-monitor-access)# source interface port-channel pc5 leaf 101</code>	Specifies the source interface port channel. (Enters the traffic direction and filter configuration, not shown here.)
Step 10	[no] shutdown Example: <code>apic1(config-monitor-access)# no shut</code>	Disables (or enables) the monitoring session.

Examples

This example shows how to configure a local access monitoring session.

```
apic1# configure terminal
apic1(config)# monitor access session mySession
apic1(config-monitor-access)# description "This is my SPAN session"
apic1(config-monitor-access)# destination interface eth 1/2 leaf 101
apic1(config-monitor-access)# source interface eth 1/1 leaf 101
apic1(config-monitor-access-source)# direction tx
apic1(config-monitor-access-source)# filter tenant t1 application appl epg epg1
apic1(config-monitor-access-source)# exit
apic1(config-monitor-access)# no shut
apic1(config-monitor-access)# show run
# Command: show running-config monitor access session mySession
# Time: Fri Nov 6 23:55:35 2015
monitor access session mySession
  description "This is my SPAN session"
  destination interface eth 1/2 leaf 101
  source interface eth 1/1 leaf 101
    direction tx
    filter tenant t1 application appl epg epg
  exit
exit
```

Configuring ERSPAN in Access Mode

In the ACI fabric, an access mode ERSPAN configuration can be used for monitoring traffic originating from access ports, port-channels, and vPCs in one or more leaf nodes.

For an ERSPAN session, the destination is always an endpoint group (EPG) which can be deployed anywhere in the fabric. The monitored traffic is forwarded to the destination wherever the EPG is moved.

SUMMARY STEPS

1. **configure**
2. **[no] monitor access session** *session-name*
3. **[no] description** *text*
4. **[no] destination tenant** *tenant-name* **application** *application-name* **epg** *epg-name* **destination-ip** *dest-ip-address* **source-ip-prefix** *src-ip-address*
5. **[no] erspan-id** *flow-id*
6. **[no] ip dscp** *dscp-code*
7. **[no] ip ttl** *ttl-value*
8. **[no] mtu** *mtu-value*
9. **exit**
10. **[no] source interface ethernet** *{[fex/]slot/port | port-range}* **leaf** *node-id*
11. **[no] source interface port-channel** *port-channel-name-list* **leaf** *node-id* [**fex** *fex-id*]
12. **[no] source interface vpc** *vpc-name-list* **leaf** *node-id1 node-id2* [**fex** *fex-id1 fex-id2*]
13. **[no] direction** *{rx | tx | both}*
14. **[no] filter tenant** *tenant-name* **application** *application-name* **epg** *epg-name*
15. **exit**

16. [no] shutdown

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] monitor access session <i>session-name</i> Example: <code>apic1(config)# monitor access session mySession</code>	Creates an access monitoring session configuration.
Step 3	[no] description <i>text</i> Example: <code>apic1(config-monitor-access)# description "This is my access ERSPAN session"</code>	Adds a description for this monitoring session. If the text includes spaces, it must be enclosed in single quotes.
Step 4	[no] destination tenant <i>tenant-name application application-name epg epg-name destination-ip dest-ip-address source-ip-prefix src-ip-address</i> Example: <code>apic1(config-monitor-access)# destination tenant t1 application appl epg epg1 destination-ip 192.0.20.123 source-ip-prefix 10.0.20.1</code>	Specifies the destination interface as a tenant and enters destination configuration mode.
Step 5	[no] erspan-id <i>flow-id</i> Example: <code>apic1(config-monitor-access-dest)# erspan-id 100</code>	Configures the ERSPAN ID for the ERSPAN session. The ERSPAN range is from 1 to 1023.
Step 6	[no] ip dscp <i>dscp-code</i> Example: <code>apic1(config-monitor-access-dest)# ip dscp 42</code>	Configures the differentiated services code point (DSCP) value of the packets in the ERSPAN traffic. The range is from 0 to 64.
Step 7	[no] ip ttl <i>ttl-value</i> Example: <code>apic1(config-monitor-access-dest)# ip ttl 16</code>	Configures the IP time-to-live (TTL) value for the ERSPAN traffic. The range is from 1 to 255.
Step 8	[no] mtu <i>mtu-value</i> Example: <code>apic1(config-monitor-access-dest)# mtu 9216</code>	Configures the maximum transmit unit (MTU) size for the ERSPAN session. The range is 64 to 9216 bytes.
Step 9	exit Example: <code>apic1(config-monitor-access-dest)#</code>	Returns to monitor access configuration mode.

	Command or Action	Purpose
Step 10	[no] source interface ethernet {[fex]/slot/port port-range} leaf node-id Example: <pre>apic1(config-monitor-access)# source interface eth 1/2 leaf 101</pre>	Specifies the source interface port or port range.
Step 11	[no] source interface port-channel port-channel-name-list leaf node-id [fex fex-id] Example: <pre>apic1(config-monitor-access)# source interface port-channel pc1 leaf 101</pre>	Specifies the source interface port-channel.
Step 12	[no] source interface vpc vpc-name-list leaf node-id1 node-id2 [fex fex-id1 fex-id2] Example: <pre>apic1(config-monitor-access)# source interface vpc pc1 leaf 101 102</pre>	Specifies the source interface vPC.
Step 13	[no] direction {rx tx both} Example: <pre>apic1(config-monitor-access-source)# direction tx</pre>	Specifies direction of traffic to be monitored. The direction can be configured independently for each source port range.
Step 14	[no] filter tenant tenant-name application application-name epg epg-name Example: <pre>apic1(config-monitor-access-source)# filter tenant t1 application appl epg epg1</pre>	Filters traffic to be monitored. The filter can be configured independently for each source port range.
Step 15	exit Example: <pre>apic1(config-monitor-access-source)# exit</pre>	Returns to access monitor session configuration mode.
Step 16	[no] shutdown Example: <pre>apic1(config-monitor-access)# no shut</pre>	Disables (or enables) the monitoring session.

Examples

This example shows how to configure an ERSPAN access monitoring session.

```
apic1# configure terminal
apic1(config)# monitor access session mySession
apic1(config-monitor-access)# description "This is my access ERSPAN session"
apic1(config-monitor-access)# destination tenant t1 application appl epg epg1 destination-ip 192.0.20.123 source-ip-prefix 10.0.20.1
apic1(config-monitor-access-dest)# erspan-id 100
```

```

apic1(config-monitor-access-dest)# ip dscp 42
apic1(config-monitor-access-dest)# ip ttl 16
apic1(config-monitor-access-dest)# mtu 9216
apic1(config-monitor-access-dest)# exit
apic1(config-monitor-access)# source interface eth 1/1 leaf 101
apic1(config-monitor-access-source)# direction tx
apic1(config-monitor-access-source)# filter tenant t1 application appl epg epg1
apic1(config-monitor-access-source)# exit
apic1(config-monitor-access)# no shut
apic1(config-monitor-access)# show run
# Command: show running-config monitor access session mySession
# Time: Fri Nov 6 23:55:35 2015
monitor access session mySession
  description "This is my ERSPAN session"
  source interface eth 1/1 leaf 101
  direction tx
  filter tenant t1 application appl epg epg1
  exit
  destination tenant t1 application appl epg epg1 destination-ip 192.0.20.123
source-ip-prefix 10.0.20.1
  ip dscp 42
  ip ttl 16
  erspan-id 9216
  mtu 9216
  exit
exit

```

This example shows how to configure a port-channel as a monitoring source.

```

apic1(config-monitor-access)# source interface port-channel pc3 leaf 105

```

This example shows how to configure a one leg of a vPC as a monitoring source.

```

apic1(config-monitor-access)# source interface port-channel vpc3 leaf 105

```

This example shows how to configure a range of ports from FEX 101 as a monitoring source.

```

apic1(config-monitor-access)# source interface eth 101/1/1-2 leaf 105

```

Configuring ERSPAN in Fabric Mode

In the ACI fabric, a fabric mode ERSPAN configuration can be used for monitoring traffic originating from one or more fabric ports in leaf or spine nodes. Local SPAN is not supported in fabric mode.

For an ERSPAN session, the destination is always an endpoint group (EPG) which can be deployed anywhere in the fabric. The monitored traffic is forwarded to the destination wherever the EPG is moved. In the fabric mode, only fabric ports are allowed as source, but both leaf and spine switches are allowed.

SUMMARY STEPS

1. **configure**
2. **[no] monitor fabric session** *session-name*
3. **[no] description** *text*

4. **[no] destination tenant** *tenant-name* **application** *application-name* **epg** *epg-name* **destination-ip** *dest-ip-address* **source-ip-prefix** *src-ip-address*
5. **[no] erspan-id** *flow-id*
6. **[no] ip dscp** *dscp-code*
7. **[no] ip ttl** *ttl-value*
8. **[no] mtu** *mtu-value*
9. **exit**
10. **[no] source interface ethernet** *{slot/port | port-range}* **switch** *node-id*
11. **[no] direction** *{rx | tx | both}*
12. **[no] filter tenant** *tenant-name* **bd** *bd-name*
13. **[no] filter tenant** *tenant-name* **vrf** *vrf-name*
14. **exit**
15. **[no] shutdown**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: <code>apic1# configure</code>	Enters global configuration mode.
Step 2	[no] monitor fabric session <i>session-name</i> Example: <code>apic1(config)# monitor fabric session mySession</code>	Creates a fabric monitoring session configuration.
Step 3	[no] description <i>text</i> Example: <code>apic1(config-monitor-fabric)# description "This is my fabric ERSPAN session"</code>	Adds a description for this monitoring session. If the text includes spaces, it must be enclosed in single quotes.
Step 4	[no] destination tenant <i>tenant-name</i> application <i>application-name</i> epg <i>epg-name</i> destination-ip <i>dest-ip-address</i> source-ip-prefix <i>src-ip-address</i> Example: <code>apic1(config-monitor-fabric)# destination tenant t1 application appl epg epg1 destination-ip 192.0.20.123 source-ip-prefix 10.0.20.1</code>	Specifies the destination interface as a tenant and enters destination configuration mode.
Step 5	[no] erspan-id <i>flow-id</i> Example: <code>apic1(config-monitor-fabric-dest)# erspan-id 100</code>	Configures the ERSPAN ID for the ERSPAN session. The ERSPAN range is from 1 to 1023.
Step 6	[no] ip dscp <i>dscp-code</i> Example: <code>apic1(config-monitor-fabric-dest)# ip dscp 42</code>	Configures the differentiated services code point (DSCP) value of the packets in the ERSPAN traffic. The range is from 0 to 64.

	Command or Action	Purpose
Step 7	[no] ip ttl <i>ttl-value</i> Example: <code>apic1(config-monitor-fabric-dest)# ip ttl 16</code>	Configures the IP time-to-live (TTL) value for the ERSPAN traffic. The range is from 1 to 255.
Step 8	[no] mtu <i>mtu-value</i> Example: <code>apic1(config-monitor-fabric-dest)# mtu 9216</code>	Configures the maximum transmit unit (MTU) size for the ERSPAN session. The range is 64 to 9216 bytes.
Step 9	exit Example: <code>apic1(config-monitor-fabric-dest)#</code>	Returns to monitor access configuration mode.
Step 10	[no] source interface ethernet {<i>slot/port</i> <i>port-range</i>} switch <i>node-id</i> Example: <code>apic1(config-monitor-fabric)# source interface eth 1/2 switch 101</code>	Specifies the source interface port or port range.
Step 11	[no] direction {<i>rx</i> <i>tx</i> <i>both</i>} Example: <code>apic1(config-monitor-fabric-source)# direction tx</code>	Specifies direction of traffic to be monitored. The direction can be configured independently for each source port range.
Step 12	[no] filter tenant <i>tenant-name</i> bd <i>bd-name</i> Example: <code>apic1(config-monitor-fabric-source)# filter tenant t1 bd bd1</code>	Filters traffic by bridge domain.
Step 13	[no] filter tenant <i>tenant-name</i> vrf <i>vrf-name</i> Example: <code>apic1(config-monitor-fabric-source)# filter tenant t1 vrf vrf1</code>	Filters traffic by VRF.
Step 14	exit Example: <code>apic1(config-monitor-fabric-source)# exit</code>	Returns to access monitor session configuration mode.
Step 15	[no] shutdown Example: <code>apic1(config-monitor-fabric)# no shut</code>	Disables (or enables) the monitoring session.

Examples

This example shows how to configure an ERSPAN fabric monitoring session.

```

apic1# configure terminal
apic1(config)# monitor fabric session mySession
apic1(config-monitor-fabric)# description "This is my fabric ERSPAN session"
apic1(config-monitor-fabric)# destination tenant t1 application appl epg epg1 destination-ip
192.0.20.123 source-ip-prefix 10.0.20.1
apic1(config-monitor-fabric-dest)# erspan-id 100
apic1(config-monitor-fabric-dest)# ip dscp 42
apic1(config-monitor-fabric-dest)# ip ttl 16
apic1(config-monitor-fabric-dest)# mtu 9216
apic1(config-monitor-fabric-dest)# exit
apic1(config-monitor-fabric)# source interface eth 1/1 switch 101
apic1(config-monitor-fabric-source)# direction tx
apic1(config-monitor-fabric-source)# filter tenant t1 bd bd1
apic1(config-monitor-fabric-source)# filter tenant t1 vrf vrf1
apic1(config-monitor-fabric-source)# exit
apic1(config-monitor-fabric)# no shut

```

Configuring ERSPAN in Tenant Mode

In the ACI fabric, a tenant mode ERSPAN configuration can be used for monitoring traffic originating from endpoint groups within a tenant.

In the tenant mode, traffic originating from a source EPG is sent to a destination EPG within the same tenant. The monitoring of traffic is not impacted if the source or destination EPG is moved within the fabric.

SUMMARY STEPS

1. **configure**
2. **[no] monitor tenant** *tenant-name* **session** *session-name*
3. **[no] description** *text*
4. **[no] destination tenant** *tenant-name* **application** *application-name* **epg** *epg-name* **destination-ip** *dest-ip-address* **source-ip-prefix** *src-ip-address*
5. **[no] erspan-id** *flow-id*
6. **[no] ip dscp** *dscp-code*
7. **[no] ip ttl** *ttl-value*
8. **[no] mtu** *mtu-value*
9. **exit**
10. **[no] source application** *application-name* **epg** *epg-name*
11. **[no] direction** {**rx** | **tx** | **both**}
12. **exit**
13. **[no] shutdown**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.

	Command or Action	Purpose
Step 2	[no] monitor tenant <i>tenant-name</i> session <i>session-name</i> Example: <code>apic1(config)# monitor tenant session mySession</code>	Creates a tenant monitoring session configuration.
Step 3	[no] description <i>text</i> Example: <code>apic1(config-monitor-tenant)# description "This is my tenant ERSPAN session"</code>	Adds a description for this access monitoring session. If the text includes spaces, it must be enclosed in single quotes.
Step 4	[no] destination tenant <i>tenant-name</i> application <i>application-name</i> epg <i>epg-name</i> destination-ip <i>dest-ip-address</i> source-ip-prefix <i>src-ip-address</i> Example: <code>apic1(config-monitor-tenant)# destination tenant t1 application app1 epg ep1 destination-ip 192.0.20.123 source-ip-prefix 10.0.20.1</code>	Specifies the destination interface as a tenant and enters destination configuration mode.
Step 5	[no] erspan-id <i>flow-id</i> Example: <code>apic1(config-monitor-tenant-dest)# erspan-id 100</code>	Configures the ERSPAN ID for the ERSPAN session. The ERSPAN range is from 1 to 1023.
Step 6	[no] ip dscp <i>dscp-code</i> Example: <code>apic1(config-monitor-tenant-dest)# ip dscp 42</code>	Configures the differentiated services code point (DSCP) value of the packets in the ERSPAN traffic. The range is from 0 to 64.
Step 7	[no] ip ttl <i>tll-value</i> Example: <code>apic1(config-monitor-tenant-dest)# ip ttl 16</code>	Configures the IP time-to-live (TTL) value for the ERSPAN traffic. The range is from 1 to 255.
Step 8	[no] mtu <i>mtu-value</i> Example: <code>apic1(config-monitor-tenant-dest)# mtu 9216</code>	Configures the maximum transmit unit (MTU) size for the ERSPAN session. The range is 64 to 9216 bytes.
Step 9	exit Example: <code>apic1(config-monitor-tenant-dest)#</code>	Returns to monitor access configuration mode.
Step 10	[no] source application <i>application-name</i> epg <i>epg-name</i> Example: <code>apic1(config-monitor-tenant)# source application app2 epg ep5</code>	Specifies the source interface port or port range.
Step 11	[no] direction { <i>rx</i> <i>tx</i> <i>both</i> } Example: <code>apic1(config-monitor-tenant-source)# direction tx</code>	Specifies direction of traffic to be monitored. The direction can be configured independently for each source port range.

	Command or Action	Purpose
Step 12	exit Example: <code>apic1(config-monitor-tenant-source)# exit</code>	Returns to access monitor session configuration mode.
Step 13	[no] shutdown Example: <code>apic1(config-monitor-tenant)# no shut</code>	Disables (or enables) the monitoring session.

Examples

This example shows how to configure an ERSPAN tenant monitoring session.

```
apic1# configure terminal
apic1(config)# monitor access session mySession
apic1(config-monitor-tenant)# description "This is my tenant ERSPAN session"
apic1(config-monitor-tenant)# destination tenant t1 application appl1 epg epg1 destination-ip
192.0.20.123 source-ip-prefix 10.0.20.1
apic1(config-monitor-tenant-dest)# erspan-id 100
apic1(config-monitor-tenant-dest)# ip dscp 42
apic1(config-monitor-tenant-dest)# ip ttl 16
apic1(config-monitor-tenant-dest)# mtu 9216
apic1(config-monitor-tenant-dest)# exit
apic1(config-monitor-tenant)# source application app2 epg epg5
apic1(config-monitor-tenant-source)# direction tx
apic1(config-monitor-tenant-source)# exit
apic1(config-monitor-tenant)# no shut
```




CHAPTER 21

Applying the show running config Output to Another Cisco APIC

This section explains how to use the `export config` and `import config` CLIs to use the `show running config` output on another Cisco APIC.

- [About Import and Export Configurations, on page 533](#)
- [Import and Export Configuration Guidelines and Limitations, on page 533](#)
- [Exporting a CLI Configuration, on page 533](#)
- [Importing a CLI Configuration, on page 534](#)

About Import and Export Configurations

The **import config** and **export config** commands enable you to apply the **show running config** output to another Cisco APIC. This section contains the guidelines for these commands and demonstrates how the commands are executed.

Import and Export Configuration Guidelines and Limitations

This section explains the guidelines and limitations for the **export config** and **import config** commands.

- Passwords and other encrypted data are not included in the configuration file.
- Some REST API configurations may not be compatible with CLI configurations; this may cause errors when applying a configuration file to a Cisco APIC.
- Some features require configurations to be in a specific order. These configurations are validated when performed through the CLI. Configurations through the REST API, however, are not validated and may cause errors when running the imported file due to missing configurations.
- Interactive commands are prefixed with a "#" and ignored when running the configuration file.

Exporting a CLI Configuration

This procedure shows how to export a configuration to a text file.

SUMMARY STEPS

1. **configure**
2. **leaf ID**
3. **interface ethernet slot/port**
4. **export-config result-file-name**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: dev4-ifc1# configure	Enters configuration mode.
Step 2	leaf ID Example: dev4-ifc1(config)# leaf 101	Identifies the leaf with the configuration to be exported.
Step 3	interface ethernet slot/port Example: dev4-ifc1(config-leaf)# interface ethernet 1/34	Identifies the slot number and port number for an existing Ethernet interface.
Step 4	export-config result-file-name Example: dev4-ifc1(config-leaf-if)# export-config /tmp/showRunnLeaf101.txt	Exports the configuration to a specified file name.

Example

This example shows how to configure export-config.

```
dev4-ifc1# config
dev4-ifc1(config)# leaf 101
dev4-ifc1(config-leaf)# interface ethernet 1/34
dev4-ifc1(config-leaf-if)# export-config /tmp/showRunnLeaf101.txt
dev4-ifc1(config-leaf-if)# cat /tmp/showRunnLeaf101.txt
config
# Command: show running-config leaf 101 interface ethernet 1 / 34
# Time: Fri Sep 23 16:03:48 2016
  leaf 101
    interface ethernet 1/34
      switchport trunk allowed vlan 602 tenant t1 external-svi l3out l3ext1sub1
    exit
  exit
dev4-ifc1(config-leaf-if)#
```

Importing a CLI Configuration

This procedure shows how to import a configuration from a text file.

SUMMARY STEPS

1. `import-config file-name`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>import-config file-name</code> Example: <pre>dev4-ifc1(config-tenant)# import-config /tmp/showRunnLeaf101.txt config # Command: show running-config leaf 101 interface ethernet 1 / 34 # Time: Fri Sep 23 16:03:48 2016 leaf 101 interface ethernet 1/34 switchport trunk allowed vlan 602 tenant t1 external-svi l3out l3extlsub1 exit exit dev4-ifc1(config)#</pre>	



CHAPTER 22

Configuring a Forwarding Scale Profile Policy

- [Forwarding Scale Profile Policy Overview, on page 537](#)
- [Supported Platforms for Forwarding Scale Profile Policies, on page 539](#)
- [Configuring the Forwarding Scale Profile Policy Using the NX-OS-Style CLI, on page 539](#)

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 APIC Release 3.2(1), depending on your TOR switch hardware, a Forwarding Scale Profile with the **High Dual Stack** option has different scales; for example:

- For Cisco Nexus 9000 Series TOR 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 TOR switches with EX in the switch name, the high dual-stack option has the same scale values as with earlier APIC releases.

See the following table for more details.

Table 12: Forwarding Scale Profile Policy Scalability

Forwarding Scale Profile Policy Options	TOR Switches with EX Names	TOR Switches with FX Names
Dual Stack	• 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	• EP MAC: 64,000 • EP IPv4: 64,000 • EP IPv6: 24,000 • LPM: 38,000 • Policy: 8,000 • Multicast: 0	• EP MAC: 64,000 • EP IPv4: 64,000 • EP IPv6: 48,000 • LPM: 38,000 • Policy: 128,000 • Multicast: 512
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	• 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**

- 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.
- 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. For a list of supported switches, see [Supported Platforms for Forwarding Scale Profile Policies, on page 539](#).
- VPCs associated with different scale profile settings are not supported. The VPC members must be configured with the same scale profile settings.

Supported Platforms for Forwarding Scale Profile Policies

The forwarding scale profile policy is only supported on the following switches:

- Cisco Nexus 9300-EX Series switches
- N9K-C9348GC-FXP
- N9K-C93108TC-FX
- N9K-C93180YC-FX

Configuring the Forwarding Scale Profile Policy Using the NX-OS-Style CLI

Before you begin

The Forwarding Scale Profile policy provides different scalability options. For more information on the scalability options, see [Forwarding Scale Profile Policy Overview, on page 537](#).

The forwarding scale profile policy requires supported switches. For a list of supported switches, see [Supported Platforms for Forwarding Scale Profile Policies, on page 539](#).

**Note**

- The switches that support the forwarding scale profile policy must be manually reloaded after the forwarding scale profile policy is applied.
- Changing the scale profile for individual members of a VPC is not allowed. If members of the same VPC are associated with different leaf profiles, then a new leaf profile should be created with both members and the scale profile applied to it.

This section demonstrates how to configure the forwarding scale profile policy using the NX-OS-style CLI.

Procedure

	Command or Action	Purpose
Step 1	configure Example: apic1# configure	Enters global configuration mode.
Step 2	no scale-profile <i>name</i> Example: apic1(config)# scale-profile testFwdScaleProf	Defines the scale-profile policy.
Step 3	profile-type {dual-stack high-dual-stack high-lpm ipv4 } Example: apic1(config-scale-profile)# profile-type ipv4	Sets the Forwarding Scale profile type.
Step 4	exit Example: apic1(config-scale-profile)# exit	Returns back to global configuration.
Step 5	template leaf-policy-group <i>leaf_group_name</i> Example: apic1(config)# template leaf-policy-group samplePolicyGrp	Defines the leaf policy group.
Step 6	scale-profile <i>name</i> Example: apic1(config-leaf-policy-group)# scale-profile testFwdScaleProf	Configures the relation between the scale-profile policy and the leaf policy group. Note The switches that support the forwarding scale profile policy must be manually reloaded after the forwarding scale profile policy is applied. For a list of switches that support the forwarding scale profile policy, see Supported Platforms for Forwarding Scale Profile Policies , on page 539.

	Command or Action	Purpose
Step 7	exit Example: <code>apicl(config-leaf-policy-group)# exit</code>	Returns back to global configuration.
Step 8	leaf-profile <i>leaf_profile_name</i> Example: <code>apicl(config)# leaf-profile sampleLeafProf</code>	Configures a leaf profile.
Step 9	leaf-group <i>leaf_group_name</i> Example: <code>apicl(config-leaf-profile)# leaf-group sampleLeafGrp</code>	Specifies a group of leaf switches.
Step 10	leaf <i>leaf_group_number</i> Example: <code>apicl(config-leaf-profile)# leaf 201</code>	Adds leaf switches to the leaf group.
Step 11	leaf-policy-group <i>leaf_policy_group_name</i> Example: <code>apicl(config-leaf-group)# leaf-policy-group samplePolicyGrp</code>	Specifies the leaf policy group to be associated to the leaf switches.
Step 12	exit Example: <code>apicl(config-leaf-policy-group)# exit</code>	Exits command mode.
Step 13	[show] running-config Example: <pre>apicl(config)# show running-config # Command: show running-config scale-profile testFwdScaleProf # Time: Thu Jul 27 22:31:29 2017 scale-profile testFwdScaleProf profile-type ipv4 exit apicl(config-scale-profile)#</pre>	Displays the current running configuration.
Step 14	[show] template leaf-policy-group Example:	Displays the current running configuration.

Examples

This example shows how to configure a the IPv4 scale profile policy.

```

apic1# configure
apic1(config)# scale-profile testFwdScaleProf
apic1(config-scale-profile)# profile-type ipv4
apic1(config-scale-profile)# exit
apic1(config)# template leaf-policy-group samplePolicyGrp
apic1(config-leaf-policy-group)# scale-profile testFwdScaleProf
apic1(config-leaf-policy-group)# exit
apic1(config)# leaf-profile sampleLeafProf
apic1(config-leaf-profile)# leaf-group sampleLeafGrp
apic1(config-leaf-profile)# leaf 201
apic1(config-leaf-group)# leaf-policy-group samplePolicyGrp
apic1(config-leaf-group)# show running-config scale-profile testFwdScaleProf
# Command: show running-config scale-profile testFwdScaleProf
# Time: Thu Jul 27 22:31:29 2017
    scale-profile testFwdScaleProf
        profile-type ipv4
        exit
apic1(config-leaf-group)# show running-config template leaf-policy-group samplePolicyGrp
# Command: show running-config template leaf-policy-group samplePolicyGrp
# Time: Tue Aug 1 11:19:44 2017
    template leaf-policy-group samplePolicyGrp
        scale-profile testFwdScaleProf
        exit
apic1(config-leaf-group)# show running-config leaf-profile sampleLeafProf
# Command: show running-config leaf-profile sampleLeafProf
# Time: Tue Aug 1 11:19:58 2017
    leaf-profile sampleLeafProf
        leaf-group sampleLeafGrp
            leaf 201
            leaf-policy-group samplePolicyGrp
        exit

```



APPENDIX **A**

Verified Scalability Using the CLI

- [CLI Scalability Limits](#), on page 543

CLI Scalability Limits

Configurable Option	Scale
Number of tenants	500
Number of Layer 3 (L3) contexts	300
Number of endpoint groups (EPGs)	3,500
Number of endpoints (EPs)	20,000
Number of bridge domains (BDs)	3,500
Number of BGP + number of OSPF sessions + EIGRP (for external connection)	300
Maximum number of vPCs	48
Maximum number of PCs, access ports	48
Maximum number of encaps per access port	1,750
Number of multicast groups	8,000
Maximum number of vzAny provided contracts	16
Maximum number of vzAny consumed contracts	16
Maximum amount of encaps per endpoint group	2 static, 1 dynamic
Security TCAM size	4,000
Number of VRFs	500
Separate-Config-Set	

Configurable Option	Scale
Tenants	100
Endpoint groups	1,000
Bridge domains	500
VRFs	100
SPAN destinations	3
NTP servers	2
Contracts	100
DNS servers	2
Syslog servers	1



APPENDIX **B**

Use Case: Three-Tier Application with Transit Topology

- [About Deploying a Three-Tier Application with Transit Topology, on page 545](#)
- [Deploying a Three-Tier Application, on page 547](#)
- [Transit Routing with OSPF and BGP, on page 549](#)

About Deploying a Three-Tier Application with Transit Topology

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.



To deploy the three-tier application, you must create the required EPGs, filters, and contracts.

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.

About Transit Routing

Transit routing enables border routers to perform bidirectional redistribution with other routing domains. Bidirectional redistribution passes routing information from one routing domain to another. Such redistribution lets the ACI fabric provide full IP connectivity between different routing domains. Doing so can also provide redundant connectivity by enabling backup paths between routing domains. For more information, see "ACI Transit Routing" in the *Cisco ACI Fundamentals Guide*.

Deploying a Three-Tier Application

Configure the tenant VRF and bridge domain.

```
apic1(config)# tenant t1
apic1(config-tenant)# vrf context v1
apic1(config-tenant-vrf)# contract enforce
apic1(config-tenant)# bridge-domain b1
apic1(config-tenant-bd)# vrf member v1
apic1(config-tenant)# interface bridge-domain b1
apic1(config-tenant-interface)# ip address 159.10.10.1/24 scope public
apic1(config-tenant-interface)# exit
```

Configure three EPGs: web, app, and db.

```
apic1(config-tenant)# application retail
apic1(config-tenant-app)# epg web
apic1(config-tenant-app-epg)# bridge-domain member b1
apic1(config-tenant-app-epg)# contract provider web
apic1(config-tenant-app-epg)# contract consumer app
apic1(config-tenant-app)# epg app
apic1(config-tenant-app-epg)# bridge-domain member b1
apic1(config-tenant-app-epg)# contract provider app
apic1(config-tenant-app-epg)# contract consumer db
apic1(config-tenant-app)# epg db
apic1(config-tenant-app-epg)# bridge-domain member b1
apic1(config-tenant-app-epg)# contract provider db
```

Configure VLAN domain.

```
apic1(config)# vlan-domain dom100
apic1(config-vlan)# vlan 100-200
```

Create port-channel and deploy the web EPG.

```
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/2-5
apic1(config-leaf-if)# channel-group po1

apic1(config-leaf)# interface port-channel po1
apic1(config-leaf-if)# vlan-domain member dom100
apic1(config-leaf-if)# switchport trunk allowed vlan 101 tenant t1 application retail epg
web
```

Create a vPC and deploy app and db EPGs.

```

apic1(config)# leaf 101,102
apic1(config-leaf)# interface ethernet 1/6,1/7
apic1(config-leaf-if)# channel-group vpc1 vpc

apic1(config)# vpc domain explicit 100 leaf 101 102
apic1(config)# vpc context leaf 101 102
apic1(config-vpc)# interface vpc vpc1
apic1(config-vpc-if)# vlan-domain member dom100
apic1(config-vpc-if)# switchport trunk allowed vlan 102 tenant t1 application retail epg
app
apic1(config-vpc-if)# switchport trunk allowed vlan 103 tenant t1 application retail epg
db

```

Configure MP-BGP.

```

apic1(config)# bgp-fabric
apic1(config-bgp-fabric)# asn 100
apic1(config-bgp-fabric)# route-reflector spine 104,105

```

Configure External-L3 EPG.

```

apic1(config-tenant)# external-l3 epg l3epg1
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# match ip 173.10.1.0/24
apic1(config-tenant-l3ext-epg)# contract consumer web

```

Configure VRF on Leaf , route-map and deploy external-L3 EPG.

```

apic1(config)# leaf 103
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# external-l3 epg l3epg1
apic1(config-leaf-vrf)# route-map map1
apic1(config-leaf-vrf-route-map)# match bridge-domain b1

```

Configure OSPF area on a sub-Interface.

```

apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant t1 vrf v1
apic1(config-leaf-ospf-vrf)# area 0.0.0.1 route-map map1 out
apic1(config-leaf)# interface ethernet 1/2
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf)# interface ethernet 1/2.150
apic1(config-leaf-if)# vrf member tenant t1 vrf v1
apic1(config-leaf-if)# ip address 169.10.10.1/24
apic1(config-leaf-if)# ip router ospf default area 0.0.0.1

```

Configure filters.

```

apic1(config-tenant)# access-list http
apic1(config-tenant-acl)# match tcp dest 80
apic1(config-tenant-acl)# match tcp dest 443

apic1(config-tenant)# access-list rmi

```



```

apic1(config-tenant-acl)# match tcp dest 1099

apic1(config-tenant)# access-list sql
apic1(config-tenant-acl)# match tcp dest 1521

```

Configure contracts.

```

apic1(config-tenant)# contract rmi
apic1(config-tenant-contract)# subject rmi
apic1(config-tenant-contract-subj)# access-group rmi both

apic1(config-tenant)# contract web
apic1(config-tenant-contract)# subject web
apic1(config-tenant-contract-subj)# access-group http both

apic1(config-tenant)# contract db
apic1(config-tenant-contract)# subject sql
apic1(config-tenant-contract-subj)# access-group sql both

```

Transit Routing with OSPF and BGP

This procedure configures transit routing between Site1 and Site2 for the three-tier application described in *Deploying a Three-Tier Application* in this chapter.

Configure External-l3 EPG (l3epg2) for Site2.

```

apic1(config-tenant)# external-l3 epg l3epg2
apic1(config-tenant-l3ext-epg)# vrf member v1
apic1(config-tenant-l3ext-epg)# match ip 174.10.1.0/24
apic1(config-tenant-l3ext-epg)# contract consumer transit
apic1(config)# leaf 102
apic1(config-leaf)# vrf context tenant t1 vrf v1
apic1(config-leaf-vrf)# external-l3 epg l3epg2

```

Configure BGP connectivity over External SVI and export route corresponding to Site1.

```

apic1(config)# leaf 102
apic1(config-leaf-vrf)# route-map map200
apic1(config-leaf-vrf-route-map)# ip prefix-list p1 match 173.10.1.0/24
apic1(config-leaf-vrf-route-map)# match prefix-list p1
apic1(config-leaf-vrf-route-map-match)# set community extended 200:1 replace

apic1(config-leaf)# interface vlan 160
apic1(config-leaf-if)# vrf member tenant t1 vrf v1
apic1(config-leaf-if)# ip address 208.1.1.2/24
apic1(config-leaf)# interface ethernet 1/11
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk allowed vlan 160 tenant t1 external-svi

apic1(config-leaf)# router bgp 100
apic1(config-bgp)# vrf member tenant t1 vrf v1
apic1(config-leaf-bgp-vrf)# neighbor 208.1.1.1
apic1(config-leaf-bgp-vrf-neighbor)# remote-as 200
apic1(config-leaf-bgp-vrf-neighbor)# update-source vlan 160

```

```
apic1(config-leaf-bgp-vrf-neighbor)# route-map map200 out
```

Configure contract provider on l3epg1 (Site1) to establish connection with l3epg2 (Site2)

```
apic1(config-tenant)# external-l3 epg l3epg1
apic1(config-tenant-l3ext-epg)# contract provider transit
```

Configure a route-map on Site1 to export the route corresponding to Site2.

```
apic1(config)# leaf 103
apic1(config-leaf-vrf)# route-map map1
apic1(config-leaf-vrf-route-map)# ip prefix-list p1 match 174.10.1.0/24
apic1(config-leaf-vrf-route-map)# match prefix-list p1
apic1(config-leaf-vrf-route-map-match)# set metric 100
```

Configure ACL and contract for transit routing.

```
apic1(config)# tenant t1
apic1(config-tenant)# access-list acl1
apic1(config-tenant-acl)# match ip
apic1(config-tenant)# contract transit
apic1(config-tenant-contract)# subject ip
apic1(config-tenant-contract-subj)# access-group acl1 both
```



APPENDIX C

Examples: Show Commands

- [Examples: Show Commands, on page 551](#)

Examples: Show Commands

show running-config

show running-config “local” to the current mode.

```
apic1(config)# leaf 103
apic1(config-leaf)# interface ethernet 1/2.150
apic1(config-leaf-if)# show running-config
# Command: show running-config leaf 103 interface ethernet 1 / 2 . 150
# Time: Tue Dec 8 08:08:37 2015
leaf 103
  interface ethernet 1/2.150
    vrf member tenant t1 vrf v1
    ip address 169.10.10.1/24
    ip router ospf default area 0.0.0.1
    exit
  exit
```

show running-config with filters.

```
apic1(config-leaf)# interface ethernet 1/2.150
apic1(config-leaf-if)# show running-config leaf 103
# Command: show running-config leaf 103
# Time: Tue Dec 8 08:10:02 2015
leaf 103
  vrf context tenant t1 vrf v1
  external-l3 ep3 l3ep3
  route-map map1
    ip prefix-list p1 permit 181.1.1.0/24
    match bridge-domain b1
    match prefix-list p1
  ...
```

show vpc, port-channel

show vpc map

```
apic1(config-leaf-if)# show vpc map
```

Legends:

N/D : Not Deployed

Virtual Port-Channel Name	Domain	VPC	Leaf Id, Name	Fex Id	PC Id	Ports
vpcl	100	1	101,leaf1		po2	eth1/6-7, eth1/40-41
vpcl	100	1	102,leaf2		po1	eth1/6-7, eth1/40-41

show port-channel map

```
apic1(config-leaf-if)# show port-channel map
```

Legends:

N/D : Not Deployed

PC: Port Channel

VPC: Virtual Port Channel

Port-Channel Name	Type	Leaf ID, Name	Fex Id	Port Channel	Ports
po1	PC	101,leaf1		po1	eth1/2-5, eth1/32-33
po1	PC	102,leaf2		po2	eth1/32-33
vpcl	VPC	101,leaf1		po2	eth1/6-7, eth1/40-41
vpcl	VPC	102,leaf2		po1	eth1/6-7, eth1/40-41

show vlan-domain

show vlan-domain name dom100

```
apic1# show vlan-domain name dom100
```

Legend:

vlanscope: L (Portlocal). Default is global

vlan-domain : dom100 Type : All

vlan : 100-200(static)

Leaf	Interface	Vlan	Type	Usage	Operational State	Operational Vlan
101	PC: po1	101	App-Epg	Tenant: t1 App: retail Epg: web	b1: down web: down	b1: vlan-18 web: vlan-21
101,102	vPC: vpcl	102	App-Epg	Tenant: t1 App: retail Epg: app	b1: down app: down	b1: vlan-18 app: vlan-19
101,102	vPC: vpcl	103	App-Epg	Tenant: t1 App: retail Epg: db	b1: down db: down	b1: vlan-18 db: vlan-20
102	eth1/11	160	Ext-svi	Tenant: t1 Vrf: v1	l2: down l3: down	vlan-18
103	eth1/2	150	Ext-subIf	Tenant: t1	-	eth1/2.14

Vrf: v1

show tenant

show tenant t1 detail

```
apic1# show tenant t1 detail
```

Detailed view for Tenant t1

Security Information:

Security Domain

VRF Information:

VRF	Policy Enforcement
-----	-----
v1	enforced

Bridge-Domain Information:

BD	VRF
-----	-----
b1	v1

Static VLAN Information:

Node	VLANs	VLAN Domains
-----	-----	-----
101	101	dom100
101 102	102,103	dom100

Static Application EPg Information:

Node	Interface	App:AEPg	BD	
Contract				
-----	-----	-----	-----	-----
101	port-channel po1	retail:web	b1	web,
				app
101 102	vpc vpc1	retail:db,retail:app	b1	app,
				db

Application EPg Information:

App:AEPg	BD
-----	-----
retail:app	b1
retail:db	b1
retail:web	b1

External L2 EPg Information:

external-l2	BD
-----	-----

External L3 EPg Information:

external-l3	VRF
-----	-----
l3epg1	v1
l3epg2	v1

show external-l3**show external-l3 interfaces**apic1# **show external-l3 interfaces**

Node	Tenant	VRF	Interface	Oper	Interface	IP Address	
102	t1	v1	vlan-160 eth1/11		vlan18 eth1/11	208.1.1.2/24	up
103	t1	v1	eth1/2.150		eth1/2.14	169.10.10.1/24	up

show external-l3 epqapic1# **show external-l3 epq**

Name	Flags	Match	Node	Entry	Oper
t1: l3epg1	vxlan: 2457600 vrf: v1 Target dscp: unspecified qosclass: unspecified Contracts	173.10.1.0/24			
t1: disabled l3epg2 disabled	vxlan: 2457600 vrf: v1 Target dscp: unspecified qosclass: unspecified Contracts	173.10.1.0/24	node-103	173.10.1.0/24	
disabled	Provided: transit Consumed: web		node-101	173.10.1.0/24	
disabled	Provided: transit Consumed: transit		node-102	173.10.1.0/24	

show external-l3 ospfapic1# **show external-l3 ospf tenant t1 vrf v1**

```

Area Id      : 0.0.0.1
Tenant       : t1
Vrf          : v1
User Config  :
Node ID      Area Properties
-----
103          Type: nssa, Cost: 1, Control: redistribute,summary

Configuration :      Operational

Node ID  Router ID  Route Map  Area Oper. Props
-----
103      10.1.0.103  map1      Type: nssa, Cost: 1, Control:
redistribute,summary, AreaId:

```

```

                                0.0.0.1
Interfaces :
  Configuration :      Operational

Node ID  Interface      IP Address      Oper. Intf  Oper. State
-----  -
103      eth1/2.150      169.10.10.1/24  eth1/2.14   down

```

show external-l3 bgp

```

apic1# show external-l3 bgp
flags_match : Properties in logical and concrete MOs are symmetric
Tenant, vrf : t1, v1

```

Node	Neighbor	Flags	RouteMap	SourceIf	Oper Peer Status
102	208.1.1.1	Allowed Self As Count: 3 TTL: 1	no (in) map200 (out)	Vlan 160	vlan18 flags_match

show external-l3 route-map

```

apic1# show external-l3 route-map
Tenant : t1 VRF: v1
Table1: Route Map Configuration

```

Node	Routemap	Type	Name	Match	Set Attributes
102	map200	PfxList	p1	100.100.100.0/24	Community
				173.10.1.0/24	
				extended:as4-nn2:200:1	
103	map1	PfxList	p1	181.1.1.0/24	Metric: 100
103	map1	BD	b1	159.10.10.1/24	

Table 2 : Route Map Usage

Node	Routemap	Protocol	Neighbors	Operational Attributes
102	map200	bgp	208.1.1.1	Pfx List: p1 100.100.100.0/24 173.10.1.0/24 ::/0
103	map1	ospf	0.0.0.1	Pfx List: p1 Metric: 100 181.1.1.0/24 ::/0

