



# Troubleshooting Cisco Application Centric Infrastructure

## Second Edition

Domenico Dastoli, Roland Ducombe, Minako Higuchi, Takuya Kishida,  
Jessica Kurtz, Joe LeBlanc, Gabriel Monroy, Austin Peacock, Pieter  
Schoenmaekers, Yuji Shimazaki, Ramses Smeyers, Joseph Young





Copyright Page	6
<b>Preface</b>	<b>9</b>
Authors	10
Acknowledgments	11
Introduction	12
<b>Fabric discovery</b>	<b>13</b>
Initial fabric setup	14
Multi-Pod discovery	37
Device replacement	65
<b>Management and core services</b>	<b>75</b>
Overview	76
In-band and out-of-band management	78
Pod Policies – BGP RR / Date&Time / SNMP	99
<b>Access policies</b>	<b>121</b>
Overview	122
Troubleshooting workflow	140
<b>Security policies</b>	<b>157</b>
Overview	158
Tools	165
EPG to EPG	170
Preferred group	183
vzAny to EPG	192

Shared L3Out to EPG	200
<b>Intra-Fabric forwarding</b>	<b>209</b>
Overview	210
Tools	213
L2 forwarding: two endpoints in same BD – no unicast routing	222
L3 forwarding: two endpoints in different BDs	249
Multi-Pod forwarding	267
Intermittent drops	297
Interface drops	305
<b>External forwarding</b>	<b>315</b>
Overview	316
Adjacencies	323
Route advertisement	341
Contract and L3Out	369
Shared L3Out	385
<b>VMM integration</b>	<b>395</b>
Overview	396
vCenter connectivity	397
Host dynamic discovery	410
Hypervisor uplink load balancing	422
<b>PBR (Policy-Based Redirect)</b>	<b>431</b>
Overview	432
Service Graph deployment	434
Forwarding	440
Other traffic flow examples	453

<b>Fabric upgrade</b>	<b>469</b>
Overview	470
Pre-upgrade validations	471
During and POST upgrade verifications	481
FPGA / EPLD / BIOS	490
CIMC	491
<b>Acronyms</b>	<b>501</b>
Acronyms	502

## Copyright Page

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

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

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

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

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

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

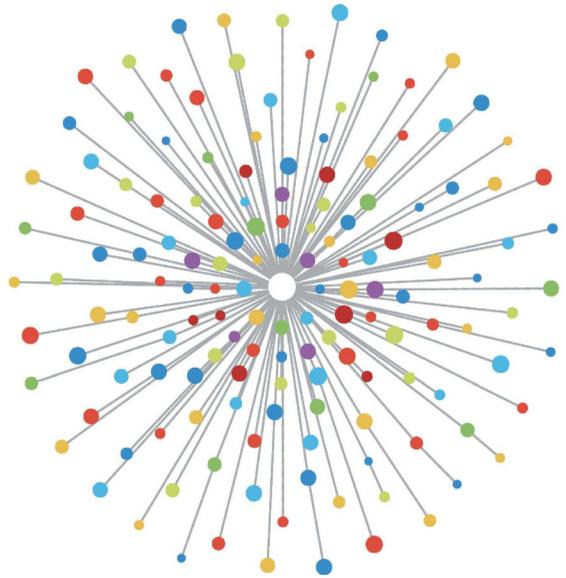
All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version. Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at [www.cisco.com/go/offices](http://www.cisco.com/go/offices).

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

© 2019 Cisco Systems, Inc. All rights reserved.



# Preface



## Authors

**Domenico Dastoli** - Cisco DCBG

**Roland Ducombe** - Cisco CX

**Minako Higuchi** - Cisco DCBG

**Takuya Kishida** - Cisco DCBG

**Jessica Kurtz** - Cisco CX

**Joe LeBlanc** - Cisco DCBG

**Gabriel Monroy** - Cisco CX

**Austin Peacock** - Cisco DCBG

**Pieter Schoenmaekers** - Cisco CX

**Yuji Shimazaki** - Cisco CX

**Ramses Smeyers** - Cisco CX

**Joseph Young** - Cisco CX

This book was written using the Book Sprints (<https://www.booksprints.net/>) method, a strongly facilitated process for collaborative authorship of books.

# Acknowledgments

While the book was produced and written by the authors, the knowledge and experience leading to it are the result of the hard work and dedication of many individuals inside Cisco who methodologically collected and wrote content since the conception and release of ACI.

Special thanks to Cisco DCBG and CX leadership teams who supported the realization of this book.

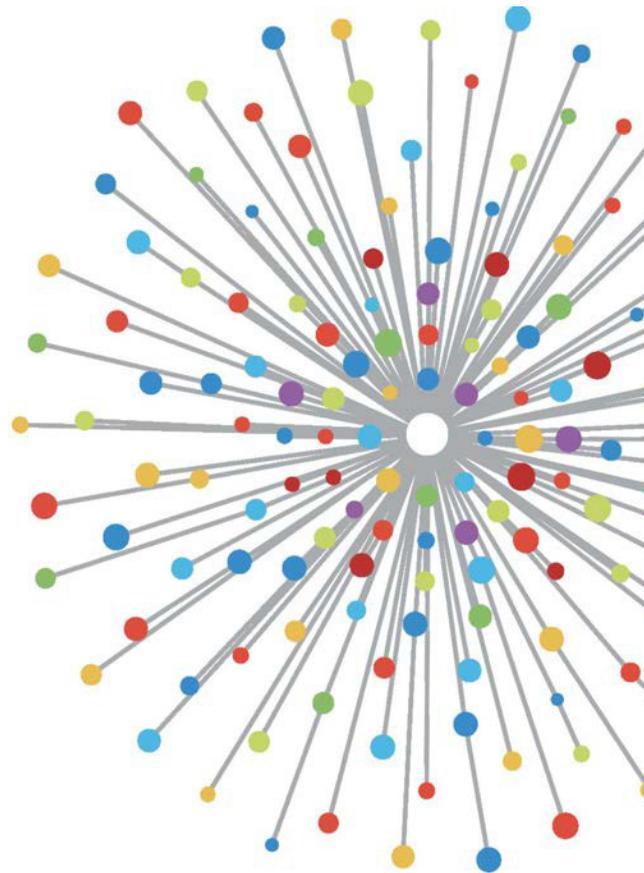
# Introduction

To keep up with the massive influx of data and increased demands on the network, networking professionals must learn to broker, connect, build, and govern not only in the datacenter, but across a vast multicloud landscape. Cisco Application Centric Infrastructure (ACI) is a major step forward in managing complexity, maximizing business benefits, and deploying workloads in any location and on any cloud.

This book systematically describes operational troubleshooting of Cisco ACI, highlighting best practices and workflows in troubleshooting an issue end-to-end. The content described not only brings together procedures from multiple sources such as Cisco Live, Cisco CX (formerly TAC) and Cisco DCBG, it also provides, where needed, an in-depth description of the discussed features to further expand the knowledge of the reader.

The authors of this book bring together 60 years of knowledge about operating and troubleshooting ACI, assuring the latest and most relevant information available to operate the ACI Infrastructure in the most efficient way. This book has been written based on Cisco ACI version 4.2 and covers the most used product and serviceability features. Most of the content, however, is applicable to older releases.

# Fabric discovery



# Initial fabric setup

## Introduction

This chapter will cover the basic steps of the fabric discovery process, common validations that can be performed during the process, and a sample of scenarios where issues occur and how to address them.

## Fabric discovery workflow

The ACI fabric discovery process follows a specific sequence of events. The basic steps are as follows.

- 1 Connect to the **KVM console** of the first APIC and complete the **setup script** by inputting values such as fabric name, APIC cluster size, and tunnel endpoint (TEP) address pool.
- 2 Once completed, APIC1 will begin sending **LLDP** via its fabric ports. The LLDP packets contain special TLVs with information such as the **infra VLAN** and its role as an APIC (also referred to as the controller).
- 3 On reception of these LLDP packets from APIC1 the leaf will program the infra VLAN on all ports where an APIC is detected.
- 4 The leaf begins sending DHCP Discovers on the now-known infra VLAN.
- 5 The user logs into the **OOB IP** of APIC1 via HTTPS and registers the first leaf node in the **Fabric Membership** submenu.
- 6 Once the leaf is given a **Node ID**, APIC1 will respond with an IP address from the configured **TEP address pool** and the DHCP process completes.
- 7 The registered leaf relays DHCP Discovers from other directly connected spines which were discovered via LLDP to APIC1.

- 8 The user will see those dynamically discovered spines appear in the Fabric Membership submenu and can register them.
- 9 Once the spines are registered, APIC1 responds with an IP address from the TEP pool and DHCP completes for those nodes.
- 10 The spines relay DHCP Discovers from all other nodes of pod1. (This is assuming there is a full-mesh between spines and leaf switches as is advised and is the typical architecture).
- 11 Once the leaf nodes connected to the other APICs are registered, the APIC cluster can be established via TCP communication amongst themselves. Make sure to complete the setup dialog on APIC2 and APIC3.
- 12 Confirm all APICs have formed a cluster and are fully fit. If this is the case, fabric discovery is complete.

Beginning in 4.2, a new CLI command is available on fabric nodes to assist in the diagnosis of common discovery issues. The following sections will cover the checks performed and provide additional validation commands to assist in troubleshooting failures.

```
leaf101# show discoveryissues
Checking the platform type.....LEAF!
Check01 - System state - in-service [ok]
Check02 - DHCP status [ok]
TEP IP: 10.0.72.67 Node Id: 101 Name: leaf101
Check03 - AV details check [ok]
Check04 - IP rechability to apic [ok]
Ping from switch to 10.0.0.1 passed
Check05 - infra VLAN received [ok]
infra vLAN:3967
Check06 - LLDP Adjacency [ok]
Found adjacency with SPINE
Found adjacency with APIC
Check07 - Switch version [ok]
version: n9000-14.2(1j) and apic version: 4.2(1j)
Check08 - FPGA/BIOS out of sync test [ok]
Check09 - SSL check [check]
SSL certificate details are valid
Check10 - Downloading policies [ok]
Check11 - Checking time [ok]
2019-09-11 07:15:53
Check12 - Checking modules, power and fans [ok]
```

### Check01 – System state

When the leaf has been allocated a Node ID and registered to the fabric, it will begin to download its bootstrap and then transition to an **in-service** state.

**Check01** - System state - out-of-service [FAIL]

**Check01** - System state - downloading-boot-script [FAIL]

To validate the current state of the leaf, the user can run `moquery -c topSystem`

```
leaf101# moquery -c topSystem
Total Objects shown: 1

# top.System
address           : 10.0.72.67
bootstrapState    : done
...
serial            : FD020160TPS
serverType        : unspecified
siteId            : 1
state            : in-service
status            :
systemUpTime      : 00:18:17:41.000
tepPool           : 10.0.0.0/16
unicastXrEpLearnDisable : no
version           : n9000-14.2(1j)
virtualMode       : no
```

### Check02 – DHCP status

**Check02** - DHCP status [FAIL]

```
ERROR: node Id not configured
ERROR: Ip not assigned by dhcp server
ERROR: Address assigner's IP not populated
TEP IP: unknown Node Id: unknown Name: unknown
```

The leaf needs to receive a TEP address via DHCP from APIC1 and then establish IP connectivity to the other APICs. The **Physical TEP** (PTEP) of the leaf is assigned to loopback0. If no address is assigned, the user can validate the leaf is sending a DHCP

Discover with tcpdump utility. Notice for this we will use interface kpm\_inb which allows you to see all CPU inband control plane network traffic.

```
(none)# tcpdump -ni kpm_inb port 67 or 68
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes
16:40:11.041148 IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from a0:36:9f:c7:a1:0c, length 300
^C
1 packets captured
1 packets received by filter
0 packets dropped by kernel
```

The user can also validate dhcpd is running on the APIC and listening on the bond0 sub-interface. The bond interface represents the fabric facing APIC ports. We will use the format bond0.<infra VLAN>.

```
apic1# ps aux | grep dhcp
root    18929  1.3  0.2 818552 288504 ?        Ssl  Sep26  87:19 /mgmt//bin/dhcpd.bin -f -4 -cf
/data//dhcp/dhcpd.conf -lf /data//dhcp/dhcpd.lease -pf /var/run//dhcpd.pid --no-pid bond0.3967
admin   22770  0.0  0.0  9108   868 pts/0    S+   19:42   0:00 grep dhcp
```

## Check03 – AV details

**Check03** - AV details check [ok]

The leaf will validate if the registered APIC has an IP in a valid range for the TEP pool. If no APIC information has been recorded yet, this check will pass. The user can see the current APIC information from the leaf node's perspective via 'acidiag avread' command. Notice in the below example that when the leaf/spine prompt is showing (none)#, this is an indication the leaf/spine is not yet a member of the fabric.

```
(none)# acidiag avread
Cluster of 0 lm(t):0(zeroTime) appliances (out of targeted 0 lm(t):0(zeroTime)) with FABRIC_DOMAIN
name=Undefined Fabric Domain Name set to version= lm(t):0(zeroTime); discoveryMode=PERMISSIVE lm(t):0(zeroTime);
drrMode=OFF lm(t):0(zeroTime)
-----
clusterTime=<diff=0 common=2019-10-01T18:51:50.315+00:00 local=2019-10-01T18:51:50.315+00:00 pf=<displForm=1
offsSt=0 offsVlu=0 lm(t):0(zeroTime)>>
-----
```

## 18 Fabric discovery

```
leaf101# acidmg avread
Cluster of 3 lm(t):0(2019-09-30T18:45:10.320-04:00) appliances (out of targeted 3 lm(t):0(2019-10-01T14:52:55.217-04:00)) with FABRIC.DOMAIN name=ACIFabric1 set to version=apic-4.2(1j) lm(t):0(2019-10-01T14:52:55.217-04:00); discoveryMode=PERMISSIVE lm(t):0(1969-12-31T20:00:00.003-04:00); drrMode=OFF
lm(t):0(1969-12-31T20:00:00.003-04:00); kafkaMode=OFF lm(t):0(1969-12-31T20:00:00.003-04:00)
  appliance id=1 address=10.0.0.1 lm(t):2(2019-09-27T17:32:08.669-04:00) tep address=10.0.0.0/16
lm(t):1(2019-07-09T19:41:24.672-04:00) routable address=192.167.0.225 lm(t):2(2019-09-30T18:37:48.916-04:00) oob
address=0.0.0.0 lm(t):0(zeroTime) version=4.2(1j) lm(t):1(2019-09-30T18:37:49.011-04:00) chassisId=c67d1076-
a2a2-11e9-874e-a390922be712 lm(t):1(2019-09-30T18:37:49.011-04:00) capabilities=0X3EEEEFFFFFFF--0X2020--0X1
lm(t):1(2019-09-26T09:32:20.747-04:00) rK=(stable,absent,0) lm(t):0(zeroTime) aK=(stable,absent,0)
lm(t):0(zeroTime) oobRk=(stable,absent,0) lm(t):0(zeroTime) oobaK=(stable,absent,0) lm(t):0(zeroTime) cntrlSbst=
(APPROVED, FCH1929V153) lm(t):1(2019-10-01T12:46:44.711-04:00) (targetMbSn= lm(t):0(zeroTime), failoverStatus=0
lm(t):0(zeroTime)) podId=1 lm(t):1(2019-09-26T09:26:49.422-04:00) commissioned=YES lm(t):101(2019-09-
30T18:45:10.320-04:00) registered=YES lm(t):3(2019-09-05T11:42:41.371-04:00) standby=NO lm(t):0(zeroTime) DRR=NO
lm(t):101(2019-09-30T18:45:10.320-04:00) apicX=NO lm(t):0(zeroTime) virtual=NO lm(t):0(zeroTime) active=YES
  appliance id=2 address=10.0.0.2 lm(t):2(2019-09-26T09:47:34.709-04:00) tep address=10.0.0.0/16
lm(t):2(2019-09-26T09:47:34.709-04:00) routable address=192.167.0.226 lm(t):2(2019-09-05T11:45:36.861-04:00) oob
address=0.0.0.0 lm(t):0(zeroTime) version=4.2(1j) lm(t):2(2019-09-30T18:37:48.913-04:00) chassisId=611febfe-
89c1-11e8-96b1-c7a7472413f2 lm(t):2(2019-09-30T18:37:48.913-04:00) capabilities=0X3EEEEFFFFFFF--0X2020--0X7
lm(t):2(2019-09-26T09:53:07.047-04:00) rK=(stable,absent,0) lm(t):0(zeroTime) aK=(stable,absent,0)
lm(t):0(zeroTime) oobRk=(stable,absent,0) lm(t):0(zeroTime) oobaK=(stable,absent,0) lm(t):0(zeroTime) cntrlSbst=
(APPROVED, FCH2045V1X2) lm(t):2(2019-10-01T12:46:44.710-04:00) (targetMbSn= lm(t):0(zeroTime), failoverStatus=0
lm(t):0(zeroTime)) podId=1 lm(t):2(2019-09-26T09:47:34.709-04:00) commissioned=YES lm(t):101(2019-09-
30T18:45:10.320-04:00) registered=YES lm(t):2(2019-09-26T09:47:34.709-04:00) standby=NO lm(t):0(zeroTime) DRR=NO
lm(t):101(2019-09-30T18:45:10.320-04:00) apicX=NO lm(t):0(zeroTime) virtual=NO lm(t):0(zeroTime) active=YES
  appliance id=3 address=10.0.0.3 lm(t):3(2019-09-26T10:12:34.114-04:00) tep address=10.0.0.0/16
lm(t):3(2019-09-05T11:42:27.199-04:00) routable address=192.167.1.163 lm(t):2(2019-10-01T13:19:08.626-04:00) oob
address=0.0.0.0 lm(t):0(zeroTime) version=4.2(1j) lm(t):3(2019-09-30T18:37:48.904-04:00) chassisId=99bade8c-
cff3-11e9-bba7-5b9066a49dc39 lm(t):3(2019-09-30T18:37:48.904-04:00) capabilities=0X3EEEEFFFFFFF--0X2020--0X4
lm(t):3(2019-09-26T10:18:13.149-04:00) rK=(stable,absent,0) lm(t):0(zeroTime) aK=(stable,absent,0)
lm(t):0(zeroTime) oobRk=(stable,absent,0) lm(t):0(zeroTime) oobaK=(stable,absent,0) lm(t):0(zeroTime) cntrlSbst=
(APPROVED, FCH1824V2VR) lm(t):3(2019-10-01T12:48:03.726-04:00) (targetMbSn= lm(t):0(zeroTime), failoverStatus=0
lm(t):0(zeroTime)) podId=2 lm(t):3(2019-09-26T10:12:34.114-04:00) commissioned=YES lm(t):101(2019-09-
30T18:45:10.320-04:00) registered=YES lm(t):2(2019-09-05T11:42:54.935-04:00) standby=NO lm(t):0(zeroTime) DRR=NO
lm(t):101(2019-09-30T18:45:10.320-04:00) apicX=NO lm(t):0(zeroTime) virtual=NO lm(t):0(zeroTime) active=YES
-----
clusterTime=<diff=15584 common=2019-10-01T14:53:01.648-04:00 local=2019-10-01T14:52:46.064-04:00 pF=<dispIForm=0
offsSt=0 offsVlu=-14400 lm(t):21(2019-09-26T10:40:35.412-04:00)>>
-----
```

### Check04 – IP reachability to APIC

When the leaf has received an IP address, it will attempt to establish TCP sessions with the APIC and begin the process of downloading its configuration. The user can validate IP connectivity to the APIC using the 'iping' utility.

```
leaf101# iping -V overlay-1 10.0.0.1
PING 10.0.0.1 (10.0.0.1) from 10.0.0.30: 56 data bytes
64 bytes from 10.0.0.1: icmp_seq=0 ttl=64 time=0.651 ms
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.474 ms
```

```
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.477 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.54 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.5 ms

--- 10.0.0.1 ping statistics --- 5 packets transmitted, 5 packets received, 0.00% packet loss
round-trip min/avg/max = 0.474/0.528/0.651 ms
```

## Check05 – Infra VLAN

```
Check05 - infra VLAN received [ok]
```

The infra VLAN check will only be successful if the node is connected to a Pod where an APIC exists. If this is not the case, the user can ignore the message because the check is expected to fail.

The leaf will determine the infra VLAN based on LLDP packets received from other ACI nodes. The first one it receives will be accepted when the switch is in discovery.

```
(none)# moquery -c lldpInst
Total Objects shown: 1

# lldp.Inst
adminSt      : enabled
childAction  :
ctrl         :
dn           : sys/lldp/inst
holdTime    : 120
infraVlan   : 3967
initDelayTime : 2
lcOwn       : local
modTs       : 2019-09-12T07:25:33.194+00:00
monPolDn    : uni/fabric/monfab-default
name        :
operErr     :
optTlvSel   : mgmt-addr,port-desc,port-vlan,sys-cap,sys-desc,sys-name
rn          : inst
status      :
sysDesc     : topology/pod-1/node-101
txFreq      : 30

bdsol-aci12-leaf1#
```

## 20 Fabric discovery

```
(none)# show vlan encap-id 3967
```

VLAN Name	Status	Ports
8 infra:default	active	Eth1/1

VLAN Type	Vlan-mode
8 enet	CE

If the infra VLAN has not been programmed on the switchport interfaces connected to the APICs, check for wiring issues detected by the leaf.

```
(none)# moquery -c lldpIf -f 'lldp.If.wiringIssues!=""'
```

```
Total Objects shown: 1
```

```
# lldp.If id          : eth1/1
adminRxSt           : enabled
adminSt             : enabled
adminTxSt           : enabled
childAction         :
descr               :
dn                  : sys/lldp/inst/if-[eth1/1]
lcOwn               : local
mac                 : E0:0E:DA:A2:F2:83
modTs               : 2019-09-30T18:45:22.323+00:00
monPolDn           : uni/fabric/monfab-default
name                :
operRxSt            : enabled
operTxSt            : enabled
portDesc            :
portMode            : normal
portVlan            : unspecified
rn                  : if-[eth1/1]
status              :
sysDesc             :
wiringIssues        : infra-vlan-mismatch
```

### Check06 – LLDP adjacency

```
Check06 - LLDP Adjacency [FAIL]
```

```
Error: leaf not connected to any spine
```

In order to determine which ports connect to other ACI devices, the leaf must receive LLDP from the other fabric nodes. To validate LLDP has been received, the user can check 'show lldp neighbors'.

```
(none)# show lldp neighbors
Capability codes:
(R) Router, (B) Bridge, (T) Telephone, (C) DOCSIS Cable Device
(W) WLAN Access Point, (P) Repeater, (S) Station, (O) Other
Device ID      Local Intf    Hold-time  Capability  Port ID
apic1          Eth1/1        120
apic2          Eth1/2        120
switch         Eth1/51       120        BR          Eth2/32
switch         Eth1/54       120        BR          Eth1/25
Total entries displayed: 4
```

## Check07 – Switch version

```
Check07 - Switch version [ok]
version: n9000-14.2(1j) and apic version: 4.2(1j)
```

If the APIC and leaf versions are not the same, fabric discovery could fail. To validate the version running on the leaf, use 'show version' or 'vsh -c 'show version'.

```
(none)# show version
Cisco Nexus Operating System (NX-OS) Software
TAC support: http://www.cisco.com/tac
Documents: http://www.cisco.com/en/US/products/ps9372/tsd_products_support_series_home.html
Copyright (c) 2002-2014, Cisco Systems, Inc. All rights reserved.
The copyrights to certain works contained in this software are
owned by other third parties and used and distributed under
license. Certain components of this software are licensed under
the GNU General Public License (GPL) version 2.0 or the GNU
Lesser General Public License (LGPL) Version 2.1. A copy of each
such license is available at
http://www.opensource.org/licenses/gpl-2.0.php and
http://www.opensource.org/licenses/lgpl-2.1.php

Software
BIOS:      version 07.66
kickstart: version 14.2(1j) [build 14.2(1j)]
  system:  version 14.2(1j) [build 14.2(1j)]
  PE:      version 4.2(1j)
BIOS compile time:      06/11/2019
kickstart image file is: /bootflash/aci-n9000-dk9.14.2.1j.bin
kickstart compile time: 09/19/2019 07:57:41 [09/19/2019 07:57:41]
system image file is:   /bootflash/auto-s
system compile time:    09/19/2019 07:57:41 [09/19/2019 07:57:41]
...
```

The same command will also work on the APICs.

```
apic1# show version
```

Role	Pod	Node	Name	Version
controller	1	1	apic1	4.2(1j)
controller	1	2	apic2	4.2(1j)
controller	2	3	apic3	4.2(1j)
leaf	1	101	leaf101	n9000-14.2(1j)
leaf	1	102	leaf102	n9000-14.2(1j)
leaf	1	103	leaf103	n9000-14.2(1j)
spine	1	1001	spine1	n9000-14.2(1j)
spine	1	1002	spine2	n9000-14.2(1j)

### Check08 – FPGA/EPLD/BIOS out of sync

The FPGA, EPLD and BIOS versions could affect the leaf node's ability to bring up the modules as expected. If these are too far out of date, the interfaces of the switch could fail to come up. The user can validate the running and expected versions of FPGA, EPLD, and BIOS with the following moquery commands.

```
(none)# moquery -c firmwareCardRunning
Total Objects shown: 2

# firmware.CardRunning
biosVer : v07.66(06/11/2019)
childAction :
descr :
dn : sys/ch/supslot-1/sup/running
expectedVer : v07.65(09/04/2018) interimVer : 14.2(1j)
internalLabel :
modTs : never
mode : normal
monPolDn : uni/fabric/monfab-default
operSt : ok
rn : running
status :
ts : 1970-01-01T00:00:00.000+00:00
type : switch
version : 14.2(1j)

# firmware.CardRunning
biosVer : v07.66(06/11/2019)
childAction :
descr :
dn : sys/ch/lcs1slot-1/lc/running
expectedVer : v07.65(09/04/2018) interimVer : 14.2(1j)
```

```

internallabel :
modTs        : never
mode         : normal
monPolDn    : uni/fabric/monfab-default
operSt       : ok
rn           : running
status       :
ts           : 1970-01-01T00:00:00.000+00:00
type         : switch
version      : 14.2(1j)

(none)# moquery -c firmwareCompRunning
Total Objects shown: 2

# firmware.CompRunning childAction :
descr      :
dn         : sys/ch/supslot-1/sup/fpga-1/running
expectedVer : 0x14 internallabel :
modTs      : never
mode       : normal
monPolDn   : uni/fabric/monfab-default
operSt     : ok
rn         : running
status     :
ts         : 1970-01-01T00:00:00.000+00:00
type       : controller
version    : 0x14

# firmware.CompRunning
childAction :
descr      :
dn         : sys/ch/supslot-1/sup/fpga-2/runnin
expectedVer : 0x4
internallabel :
modTs      : never
mode       : normal
monPolDn   : uni/fabric/monfab-default
operSt     : ok
rn         : running
status     :
ts         : 1970-01-01T00:00:00.000+00:00
type       : controller
version    : 0x4

```

If the running FPGA version does not match the expected FPGA version, it can be updated with the steps found in the chapter "Fabric discovery", section "Device replacement" under scenario "Leaf/Spine EPLD/FPGA not correct, F1582".

## Check09 – SSL check

```
Check09 - SSL check [check]
SSL certificate details are valid
```

SSL communication is used between all fabric nodes to ensure encryption of control plane traffic. The SSL certificate used is installed during manufacturing and is generated based on the serial number of the chassis. The format of the subject should be as follows:

```
subject= /serialNumber=PID:N9K-C93xxxxx SN:FD0xxxxxxx/CN=FD0xxxxxxx
```

To validate SSL certificate during the discovery of a switch, use the following command.

```
(none)# cd /securedata/ssl && openssl x509 -noout -subject -in server.crt
subject= /serialNumber=PID:N9K-C93180YC-EX SN:FD020432LH1/CN=FD020432LH1
```

Note that the above will only work as non-root user if the switch node is still in discovery.

The chassis serial number can be found with the following command.

```
(none)# show inventory
NAME: "Chassis", DESCR: "Nexus C93180YC-EX Chassis"
PID: N9K-C93180YC-EX , VID: V00 , SN: FD020160TPS
...
```

Additionally, the certificate must be valid at the current time. To view the valid dates of the certificate, use the '-dates' flag in the openssl command.

```
(none)# cd /securedata/ssl && openssl x509 -noout -dates -in server.crt
notBefore=Nov 28 17:17:05 2016 GMT
notAfter=Nov 28 17:27:05 2026 GMT
```

## Check10 – Download policy

```

Check10 - Downloading policies [FAIL]
Registration to all PM shards is not complete
Policy download is not complete

```

Once the leaf has IP reachability to the APIC, it will download its configuration from the APIC and the APIC will acknowledge that the download is complete. The status of this process can be viewed with the following command.

```

(none)# moquery -c pconsBootStrap
Total Objects shown: 1

# pcons.BootStrap
allLeaderAked      : no
allPortsInService  : yes
allResponsesFromLeader : yes
canBringPortInService : no
childAction        :
completedPolRes   : no
dn                 : rescont/bootstrap
lcOwn              : local
modTs              : 2019-09-27T22:52:48.729+00:00
rn                 : bootstrap
state              : completed
status             :
timerTicks         : 360
try                : 0
worstCaseTaskTry   : 0

```

## Check11 – Time

```

Check11 - Checking time [ok]
2019-10-01 17:02:34

```

This check shows the user the current time. If there is too much delta between APIC and switch time, discovery could fail. On the APIC, the time can be checked with the date command.

```
apic1# date
Tue Oct 1 14:35:38 UTC 2019
```

### Check12 – Module, PSU, fan check

For the switch to have connectivity to other devices, the modules need to be up and online. This can be validated via 'show module' and 'show environment' commands.

```
(none)# show module

Mod  Ports  Module-Type          Model          Status
-----
1    54     48x10/25G+6x40/100G Switch  N9K-C93180YC-EX  ok

Mod  Sw          Hw
-----
1    14.2(1j)    0.3050

Mod  MAC-Address(es)          Serial-Num
-----
1    e0-0e-da-a2-f2-83 to e0-0e-da-a2-f2-cb  FD020160TPS

Mod  Online Diag Status
-----
1    pass
```

```
(none)# show environment
Power Supply:
Voltage: 12.0 Volts

Power Supply      Model          Actual Output (Watts)  Total Capacity (Watts)  Status
-----
1    NXA-PAC-650W-PI  0 W                650 W                shut
2    NXA-PAC-650W-PI  171 W              650 W                ok

Module           Model          Actual Power Draw (Watts)  Allocated (Watts)  Status
-----
1    N9K-C93180YC-EX  171 W              492 W                Powered-Up
fan1  NXA-FAN-30CFM-B  N/A                N/A                  Powered-Up
fan2  NXA-FAN-30CFM-B  N/A                N/A                  Powered-Up
fan3  NXA-FAN-30CFM-B  N/A                N/A                  Powered-Up
fan4  NXA-FAN-30CFM-B  N/A                N/A                  Powered-Up
```

N/A - Per module power not available

Power Usage Summary:

```

-----
Power Supply redundancy mode (configured)           Non-Redundant(combined)
Power Supply redundancy mode (operational)         Non-Redundant(combined)

Total Power Capacity (based on configured mode)    650 W
Total Power of all Inputs (cumulative)             650 W
Total Power Output (actual draw)                   171 W
Total Power Allocated (budget)                     N/A
Total Power Available for additional modules       N/A

Fan:
-----
Fan          Model          Hw          Status
-----
Fan1(sys_fan1)  NXA-FAN-30CFM-B  --          ok
Fan2(sys_fan2)  NXA-FAN-30CFM-B  --          ok
Fan3(sys_fan3)  NXA-FAN-30CFM-B  --          ok
Fan4(sys_fan4)  NXA-FAN-30CFM-B  --          ok
Fan_in_PS1     --                --          unknown
Fan_in_PS2     --                --          ok
Fan Speed: Zone 1: 0x7f
Fan Air Filter : Absent

Temperature:
-----
Module  Sensor                MajorThresh  MinorThres  CurTemp  Status
        (Celsius)         (Celsius)   (Celsius)
-----
1      Inlet(1)                70           42           35      normal
1      outlet(2)              80           70           37      normal
1      x86 processor(3)      90           80           38      normal
1      Sugarbowl(4)          110          90           60      normal
1      Sugarbowl1 vrm(5)    120          110          50      normal

```

If a module is not coming online, reseal the module and check for FPGA, EPLD, or BIOS mismatches.

## Broken scenarios

### First leaf does not appear in Fabric Membership

In this scenario, the user logs into APIC1 after completing the setup script and no switches have appeared in Fabric Membership. For the discovery of first leaf to occur successfully, the APIC should receive a DHCP Discover from the leaf in discovery phase.

Check that APIC1 is sending LLDP TLVs matching the parameters set in the setup script.

```
apic1# acdiag run lldptool out eth2-1
Chassis ID TLV
  MAC: e8:65:49:54:88:a1
Port ID TLV
  MAC: e8:65:49:54:88:a1
Time to Live TLV
  120
Port Description TLV
  eth2-1
System Name TLV
  apic1
System Description TLV
  topology/pod-1/node-1
Management Address TLV
  IPv4: 10.0.0.1
  Ifindex: 4
Cisco Port State TLV
  1
Cisco Node Role TLV
  0
Cisco Node ID TLV
  1
Cisco POD ID TLV
  1
Cisco Fabric Name TLV
  ACIFabric1
Cisco Appliance Vector TLV
  Id: 1
  IPv4: 10.0.0.1
  UUID: c67d1076-a2a2-11e9-874e-a390922be712
Cisco Node IP TLV
  IPv4:10.0.0.1
Cisco Port Role TLV
  2
Cisco Infra VLAN TLV
  3967
Cisco Serial Number TLV
  FCH1929V153
Cisco Authentication Cookie TLV
  1372058352
Cisco Standby APIC TLV
  0
End of LLDPDU TLV
```

Also validate that APIC1 is receiving LLDP from the directly connected leaf node.

```
apic1# acdiag run lldptool in eth2-1
Chassis ID TLV
  MAC: e0:0e:da:a2:f2:83
Port ID TLV
  Local: Eth1/1
Time to Live TLV
  120
Port Description TLV
  Ethernet1/1
System Name TLV
  switch
System Description TLV
  Cisco Nexus Operating System (NX-OS) Software 14.2(1j)
TAC support: http://www.cisco.com/tac
Copyright (c) 2002-2020, Cisco Systems, Inc. All rights reserved.
System Capabilities TLV
  System capabilities: Bridge, Router
  Enabled capabilities: Bridge, Router
Management Address TLV
  MAC: e0:0e:da:a2:f2:83
  Ifindex: 83886080
Cisco 4-wire Power-via-MDI TLV
  4-Pair PoE supported
  Spare pair Detection/Classification not required
  PD Spare pair Desired State: Disabled
  PSE Spare pair Operational State: Disabled
Cisco Port Mode TLV
  0
Cisco Port State TLV
  1
Cisco Serial Number TLV
  FD020160TPS
Cisco Model TLV
  N9K-C93180YC-EX
Cisco Firmware Version TLV
  n9000-14.2(1j)
Cisco Node Role TLV
  1
Cisco Infra VLAN TLV
  3967
Cisco Node ID TLV
  0
End of LLDPDU TLV
```

## 30 Fabric discovery

If APIC1 is receiving LLDP from the directly connected leaf node, the leaf should program the infra VLAN on the ports connected to the APIC. This VLAN programming can be validated via the 'show vlan encap-id <x>' command where 'x' is the configured infra VLAN.

```
(none)# show vlan encap-id 3967
VLAN Name                Status    Ports
-----
8   infra:default          active   Eth1/1

VLAN Type  Vlan-mode
-----
8   enet    CE
```

If the infra VLAN has not been programmed, check for wiring issues detected by the leaf node.

```
(none)# moquery -c lldpIf -f 'lldp.If.wiringIssues!=""'
Total Objects shown: 1

# lldp.If
id          : eth1/1
adminRxSt  : enabled
adminSt    : enabled
adminTxSt  : enabled
childAction :
descr      :
dn         : sys/lldp/inst/if-[eth1/1]
lcOwn      : local
mac        : E0:0E:DA:A2:F2:83
modTs      : 2019-09-30T18:45:22.323+00:00
monPolDn   : uni/fabric/monfab-default
name       :
operRxSt   : enabled
operTxSt   : enabled
portDesc   :
portMode   : normal
portVlan   : unspecified
rn         : if-[eth1/1]
status     :
sysDesc    :
wiringIssues : infra-vlan-mismatch
```

When wiring issues attribute is set to 'infra-vlan-mismatch', the indication is that the leaf has learned of a different infra VLAN than the value which the APIC is sending (the

APIC sent value can be verified using the command 'moquery -c lldpInst'). This scenario can occur if the leaf receives LLDP from a node that was once a part of another fabric. Essentially, a node in discovery will accept the first infra VLAN received via LLDP. To resolve this, remove the connections between this leaf and the other ACI nodes, except for the APIC, then clean reload the switch with 'acdiag touch clean' and 'reload' commands. Once the switch has booted, verify the correct infra VLAN is programmed. If this is true, connections can be restored to the other nodes and the user can proceed further with the ACI fabric set up.

### Other APICs do not join the cluster

In this scenario, all fabric nodes have been discovered but APIC2 and 3 have not yet joined the APIC cluster.

Validate the setup script values across APICs. Values that must match are:

- Fabric domain
- Fabric ID
- TEP pool
- Infra VLAN
- GIPo
- Cluster size
- Firmware version

```
apic1# cat /data/data_admin/sam_exported.config
Setup for Active and Standby APIC

fabricDomain = ACIFabric1
fabricID = 1
systemName =apic1
controllerID = 1
tepPool = 10.0.0.0/16
infraVlan = 3967
GIPo = 225.0.0.0/15
clusterSize = 3
standbyApic = NO
enableIPv4 = Y
```

## 32 Fabric discovery

```
enableIPv6 = N
firmwareVersion = 4.2(1j)
ifcIpAddr = 10.0.0.1
apicX = NO
podId = 1
oobIpAddr = 10.48.22.69/24
```

Verify common issues with 'acidiag cluster' command on all 3 APICs.

```
apic1# acidiag cluster
Admin password:

Product-name = APIC-SERVER-M1
Serial-number = FCH1906V1XV
Running...

Checking Core Generation: OK
Checking Wiring and UUID: OK
Checking AD Processes: Running
Checking All Apics in Commission State: OK
Checking All Apics in Active State: OK
Checking Fabric Nodes: OK
Checking Apic Fully-Fit: OK
Checking Shard Convergence: OK
Checking Leadership Degration: Optimal leader for all shards
Ping OOB IPs:
APIC-1: 10.48.22.69 - OK
APIC-2: 10.48.22.70 - OK
APIC-3: 10.48.22.71 - OK
Ping Infra IPs:
APIC-1: 10.0.0.1 - OK
APIC-2: 10.0.0.2 - OK
APIC-3: 10.0.0.3 - OK
Checking APIC Versions: Same (4.2(1j))
Checking SSL: OK

Done!
```

Finally, use 'avread' to validate if these settings match across all APICs. Note that this is a different command from the typical 'acidiag avread' which shows similar output, but it is parsed for easier consumption.

```

apic1# avread
Cluster:
-----
fabricDomainName    ACIFabric1
discoveryMode       PERMISSIVE
clusterSize         3
version             4.2(1j)
drrMode             OFF
operSize            3

APICs:
-----
version             APIC 1          APIC 2          APIC 3
address            10.0.0.1        10.0.0.2        10.0.0.3
oobAddress         10.48.22.69/24 10.48.22.70/24 10.48.22.71/24
routableAddress    0.0.0.0         0.0.0.0         0.0.0.0
tepAddress         10.0.0.0/16     10.0.0.0/16     10.0.0.0/16
podId              1                1                1
chassisId          3c9e5024-.-5a78727f 573e12c0-.-6b8da0e5 44c4bf18-.-20b4f528  cntrlSbst_serial
(APPROVED,FCH1906V1XV) (APPROVED,FCH1921V1Q9) (APPROVED,FCH1906V1PW)
active             YES              YES              YES
flags              cra-             cra-             cra-
health             255             255             255
apic1#

```

### Spine does not appear in Fabric Membership

In this scenario, the first leaf has been discovered in the fabric but no spines have appeared for discovery under the Fabric Membership submenu.

Validate physical connectivity from leaf to spine. In the example below, the leaf switch is connected to a spine via interface e1/49.

```

leaf101# show int eth1/49
Ethernet1/49 is up
admin state is up, Dedicated Interface
Hardware: 1000/10000/100000/40000 Ethernet, address: 0000.0000.0000 (bia e00e.daa2.f3f3)
MTU 9366 bytes, BW 1000000000 Kbit, DLY 1 usec
reliability 255/255, txload 1/255, rxload 1/255
Encapsulation ARPA, medium is broadcast
Port mode is routed
full-duplex, 100 Gb/s
...

```

## 34 Fabric discovery

If the port is in an **out-of-service** status, check on the spine that LLDP has been received from the directly connected leaf.

```
(none)# show lldp neighbors
Capability codes:
(R) Router, (B) Bridge, (T) Telephone, (C) DOCSIS Cable Device
(W) WLAN Access Point, (P) Repeater, (S) Station, (O) Other
Device ID      Local Intf    Hold-time  Capability  Port ID
leaf102        Eth2/27      120        BR           Eth1/53
leaf103        Eth2/29     120       BR          Eth1/49
leaf101        Eth2/32      120        BR           Eth1/51
Total entries displayed: 3
```

Another validation is to verify that there is no version difference between leaf and spine. If there is, remediate the situation by copying the newer version to /bootflash of the spine. Then, configure the switch to boot to the software with the following commands:

```
(none)# ls -alh /bootflash
total 3.0G
drwxrwxr-x 3 root admin 4.0K Oct  1 20:21 .
drwxr-xr-x 50 root root  1.3K Oct  1 00:22 ..
-rw-r--r-- 1 root root  3.5M Sep 30 21:24 CpuUsage.Log
-rw-rw-rw- 1 root root  1.7G Sep 27 14:50 aci-n9000-dk9.14.2.1j.bin
-rw-r--r-- 1 root root  1.4G Sep 27 21:20 auto-s
-rw-rw-rw- 1 root root    2 Sep 27 21:25 diag_bootup
-rw-r--r-- 1 root root   54 Oct  1 20:20 disk_log.txt
-rw-rw-rw- 1 root root  693 Sep 27 21:23 libmon.logs
drwxr-xr-x 4 root root  4.0K Sep 26 15:24 lxc
-rw-r--r-- 1 root root 384K Oct  1 20:20 mem_log.txt
-rw-r--r-- 1 root root 915K Sep 27 21:10 mem_log.txt.old.gz
-rw-rw-rw- 1 root root  12K Sep 27 21:17 urib_api_log.txt
```

```
(none)# setup-bootvars.sh aci-n9000-dk9.14.2.1j.bin
In progress
In progress
In progress
In progress
Done
```

If the new image is continuously removed from bootflash, ensure that the folder is less than half full by removing older images or auto-s file; check the space utilization by using 'df -h' on the switch.

After setting the boot variable, reload the switch and it should boot to the new version.

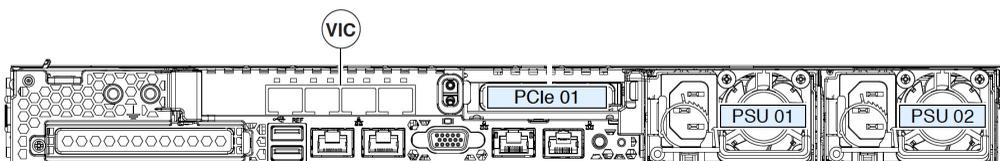
FPGA, EPLD, and BIOS validation might be required after the reload. Please refer to the sub-section "Leaf/Spine EPLD/FPGA not correct, F1582" for further troubleshooting on this matter.

### After initial fabric discovery, cluster is flapping between fully-fit and degraded

If this is happening after a new fabric setup, it can be caused by incorrect cabling of the APIC-M3 or APIC-L3 connecting into the fabric. You can confirm such incorrect cabling by executing "show lldp neighbors" on both leaf switches connected to the APIC. You will notice after executing this multiple times that both leaf switches are seeing the same APIC interface.

The back of an APIC-M3/L3 server looks like the following:

Rear-view of APIC-M3/L3 server



Note that for an APIC-M3/L3, the VIC card has 4 ports: ETH2-1, ETH2-2, ETH2-3, and ETH2-4, as seen below:

View of APIC VIC 1455 with labels

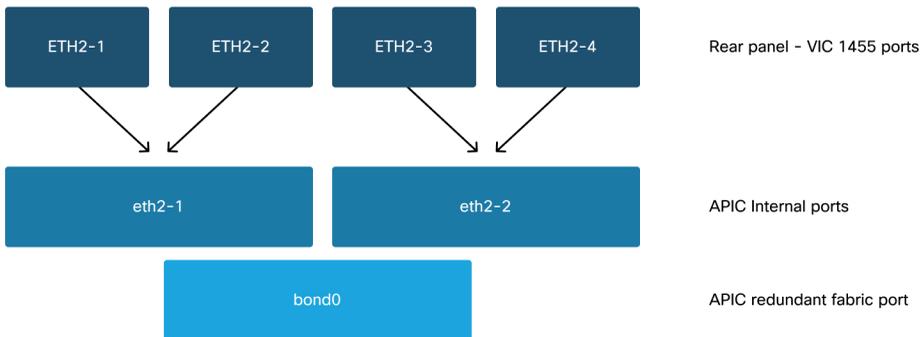


The rules to connect the APIC server to leaf switches are as follows:

- All ports must have the same speed, either 10-Gigabit or 25-Gigabit.
- ETH2-1 and ETH2-2 is one port-channel pair, corresponding to eth2-1 ('ifconfig' output) from the APIC OS.
- ETH2-3 and ETH2-4 is the other port-channel pair, corresponding to eth2-2 ('ifconfig' output) on APIC OS.
- Only one connection is allowed per port-channel pair. For example, connect one cable to either ETH2-1 or ETH2-2, and connect another cable to either ETH2-3 or ETH2-4 (**Never connect both ETHs in a port channel pair. This will lead to fabric discovery issues.**).

For further understanding, the following is a representation of the VIC port mapping to APIC bond.

#### VIC 1455 ports – APIC redundant fabric port



# Multi-Pod discovery

## Overview

ACI Multi-Pod allows for the deployment of a single APIC cluster to manage multiple ACI networks that are interconnected. Those separate ACI networks are called 'Pods' and each Pod is a regular two or three-tier spine-leaf topology. A single APIC cluster can manage several Pods.

A Multi-Pod design also allows for the extension of ACI fabric policies across Pods that can physically exist in multiple rooms or even across remote datacenter locations. In a Multi-Pod design, any policy defined on the APIC controller cluster is automatically made available to all Pods.

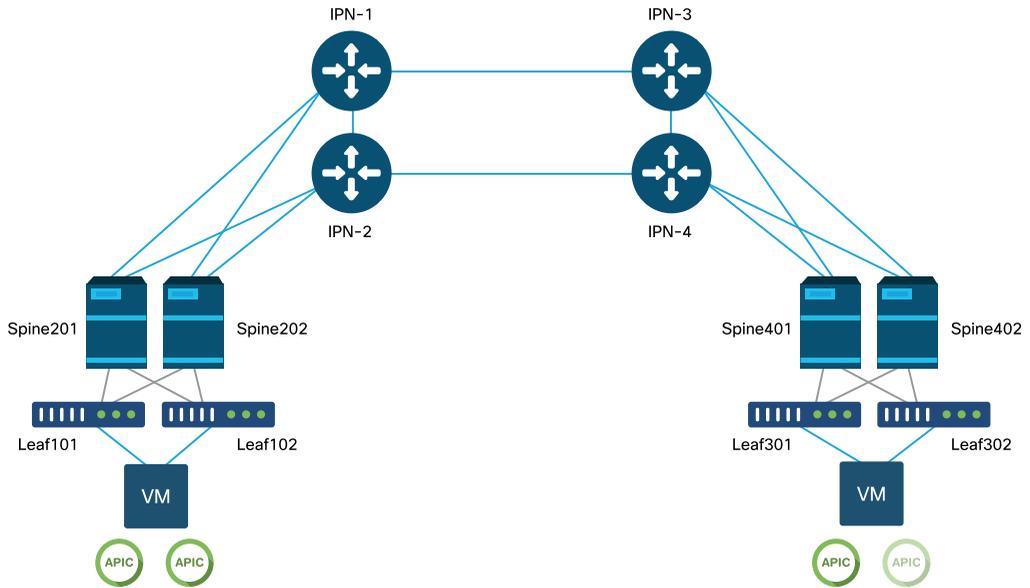
Finally, a Multi-Pod design increases failure domain isolation. In fact, each Pod runs its own instance of COOP, MP-BGP and IS-IS protocol so faults and issues with any of these protocols are contained within that Pod and cannot spread to other Pods.

Please refer to the document "ACI Multi-Pod White Paper" on [cisco.com](http://cisco.com) for more information on Multi-Pod design and best practices.

The main elements of a Multi-Pod ACI fabric are the leaf and spine switches, the APIC controllers and the IPN devices.

This example dives into the troubleshooting workflow for issues related to setting up an ACI Multi-Pod fabric. The reference topology used for this section is depicted in the picture below:

ACI Multi-Pod reference topology



## Troubleshooting workflow

### Verify ACI policies

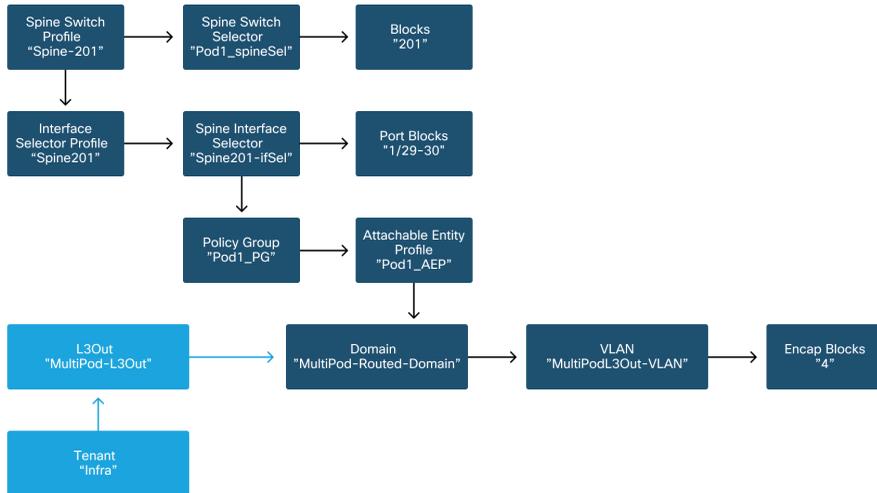
#### Access Policies

Multi-Pod uses an L3Out in order to connect Pods via the 'infra' tenant. This means the standard set of access policies need to be in place to activate the required Multi-Pod L3Out encapsulation (VLAN-4) on the spine ports facing towards the IPN.

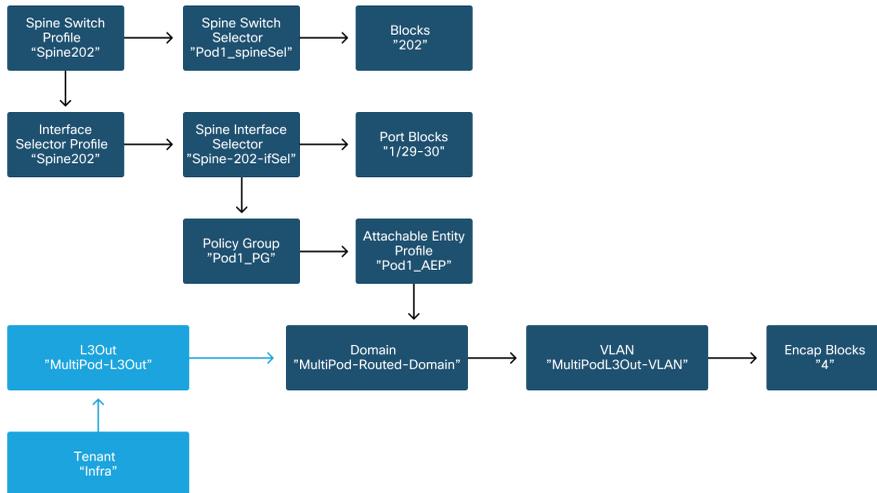
Access Policies can be configured through the 'Add Pod' wizard which should be used to deploy Multi-Pod. After using the wizard, deployed policy can be verified from the APIC GUI. If policies are not properly configured, a fault will appear on the infra tenant and connectivity from spines to the IPN may be not working as expected.

The following schemas can be referenced while verifying access policy definition for the IPN-facing interfaces on the spine nodes:

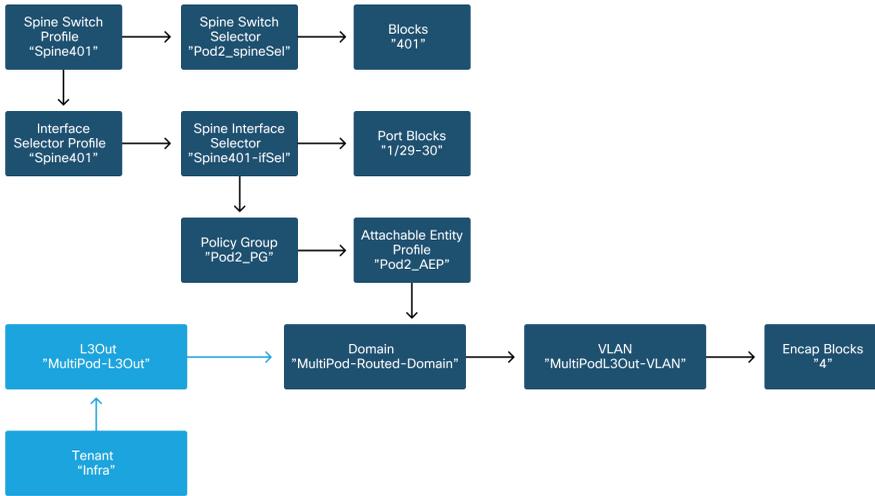
### Spine201



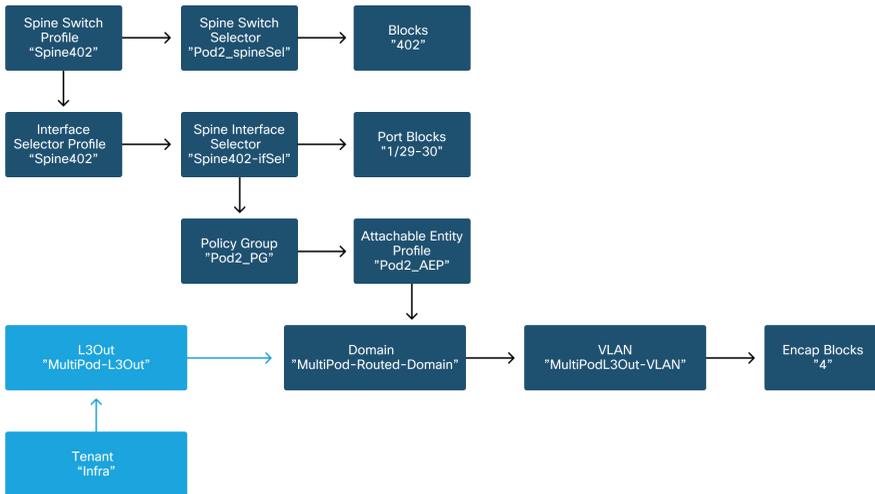
### Spine202



### Spine401

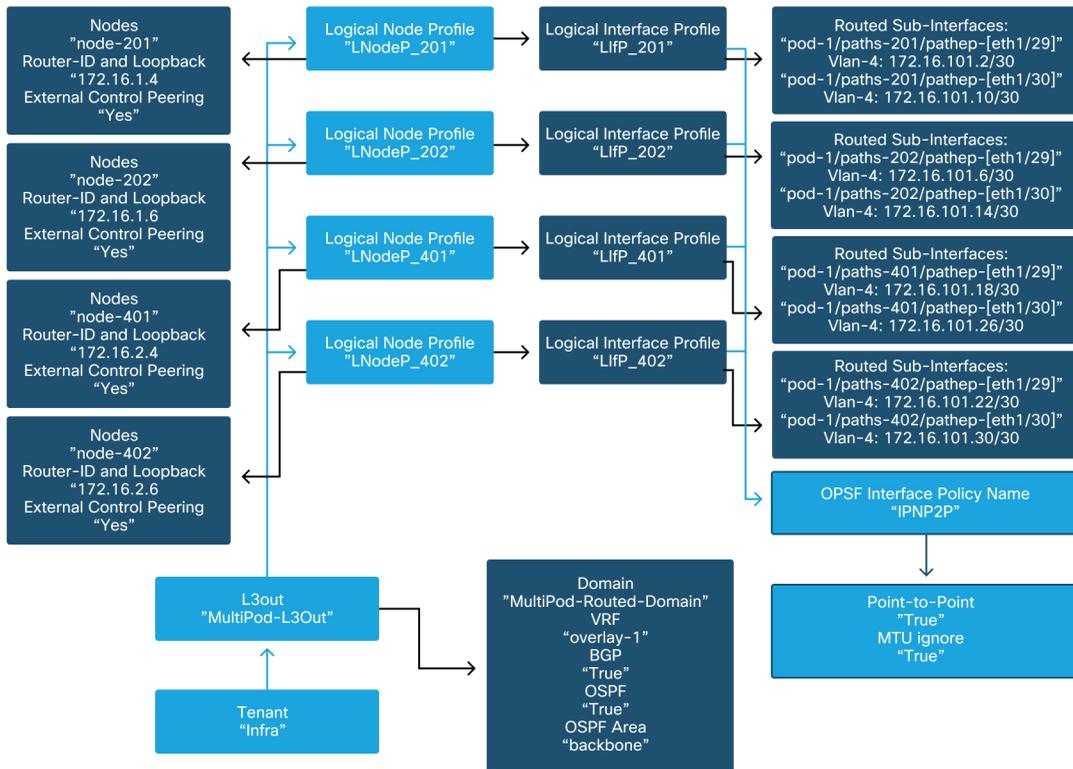


### Spine402



In the infra tenant, the Multi-Pod L3Out should be configured as per the following schema:

Multi-Pod L3Out in infra tenant



Below is a reference shot of the Multi-Pod L3Out Logical Interface Profile configuration. The router sub-interface definitions should look like the picture below for spine 201

### Logical Interface Profile in infra L3Out

Logical Interface Profile - LIFP\_201

Policy Faults History

General Routed Sub-Interfaces

Path	IP Address	Secondary IP Address	MAC Address	MTU (bytes)	Encap
Pod-1/Node-201/eth1/29	172.16.101.2/30		00:22:B...	9150	vlan-4
Pod-1/Node-201/eth1/30	172.16.101.10/30		00:22:B...	9150	vlan-4

Show Usage Reset Submit

## 44 Fabric discovery

For each Pod, there should be a TEP Pool defined as in the picture below. Note that the TEP Pool will be used from APIC controller to provision the IP addresses of the nodes for the overlay-1 VRF.

### Pod Fabric Setup Policy

The screenshot shows the APIC interface for configuring Pod Fabric Setup Policies. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Fabric' menu item is highlighted. The left sidebar shows the 'Inventory' section with 'Pod Fabric Setup Policy' highlighted. The main content area is titled 'Pod Fabric Setup Policy' and has tabs for 'Physical Pods' and 'Virtual Pods'. Below the tabs is a table with the following data:

Pod ID	TEP Pool	Remote ID
1	10.0.0.0/16	
2	10.1.0.0/16	

## Fabric External Connection Policy default

Verify that in the infra tenant the 'Fabric Ext Policy default' object is defined and configured appropriately. A sample of this configuration is shown in the figures below.

### Fabric External Connection Policy default

The screenshot displays the Cisco APIC web interface. The top navigation bar includes the Cisco logo, the text 'APIC', and the user 'admin'. Below this is a menu with 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Tenants' menu is expanded, showing 'ALL TENANTS', 'Add Tenant', and a search bar. The 'infra' tenant is selected, and its configuration tree is visible on the left. The tree includes 'Quick Start', 'infra', 'Application Profiles', 'Networking', 'Contracts', 'Policies', 'Protocol', 'BFD', 'BGP', 'Custom QOS', 'DHCP', 'DSCP class-cos translation policy fo...', 'Data Plane Policing', 'EIGRP', 'End Point Retention', 'Fabric Ext Connection Policies', and 'Fabric Ext Connection Policy defa...'. The 'Fabric Ext Connection Policies' folder is expanded, showing the 'Fabric Ext Connection Policy default' object. The main content area displays the configuration for this object, titled 'Intrasite/Intersite Profile - Fabric Ext Connection Policy default'. The configuration includes a 'Policy' tab, 'Properties' section with 'Fabric ID: 1', 'Name: default', and 'Community: extended:as2-nn4:5:16', 'Enable Pod Peering Profile' checked, 'Pod Peering Profile' section with 'Peering Type' set to 'Full Mesh', and 'Pod Connection Profile' section. The configuration is saved with 'Show Usage', 'Reset', and 'Submit' buttons.

## Dataplane TEP

The screenshot shows the Cisco APIC interface. At the top, the navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Tenants' tab is active, and the 'infra' tenant is selected in the breadcrumb trail. The left sidebar shows a tree view of configuration objects, with 'Fabric Ext Connection Policies' and 'Fabric Ext Connection Policy default' highlighted. The main content area displays the configuration for 'Intrasite/Intersite Profile - Fabric Ext Connection Policy default'. It features a table for 'Data Plane TEP' and a section for 'Fabric External Routing Profile'.

Pod ID	Data Plane TEP	Multi-site Unicast Data Plane TEP
1	172.16.1.1/32	
2	172.16.2.1/32	

Name	Subnet
multiPodL3Out_RoutingProfile	172.16.101.10/30, 172.16.101.14/30, 172...

## Fabric External Routing Profile subnets

### Fabric External Routing Profile - multipodL3Out\_RoutingProfile

The screenshot shows a configuration window for the Fabric External Routing Profile named 'multipodL3Out\_RoutingProfile'. The window has a title bar with standard OS icons and a refresh button. Below the title bar, there are three tabs: 'Profile' (selected), 'Faults', and 'History'. The main content area is titled 'Properties' and contains the following fields:

- Name:** multipodL3Out\_RoutingProfile
- Description:** optional (with an empty text input field below it)
- Subnet Addresses:** A list of subnets, each with a trash icon and a plus sign for removal and addition respectively.

The subnets listed are:

Subnet
172.16.101.10/30
172.16.101.14/30
172.16.101.18/30
172.16.101.2/30
172.16.101.22/30
172.16.101.26/30
172.16.101.30/30
172.16.101.6/30

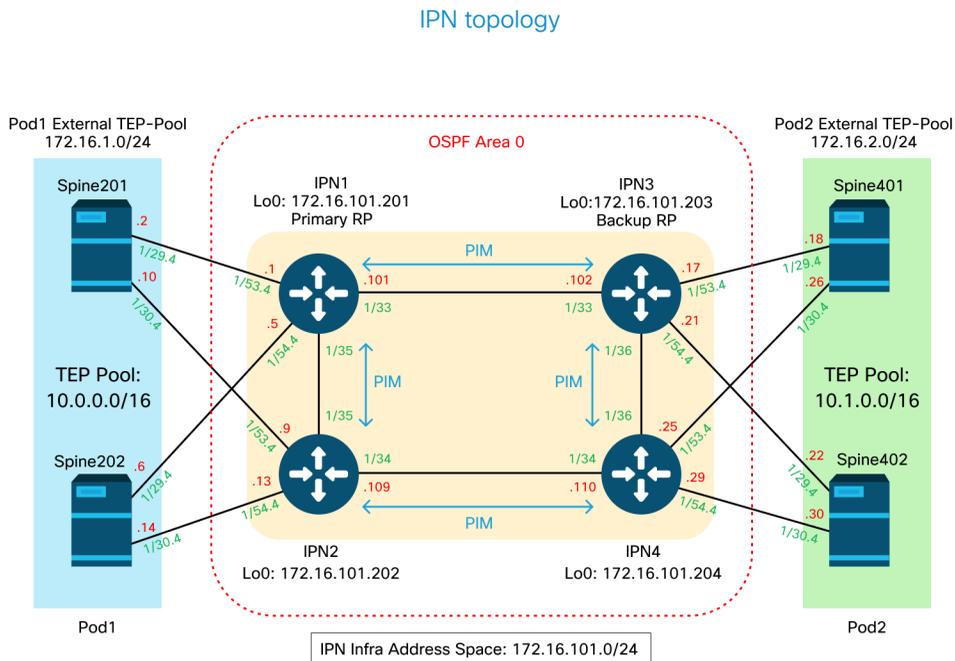
At the bottom right of the window, there are three buttons: 'Show Usage', 'Close', and 'Submit'.

The Fabric External Routing Profile enables the user to verify whether all routed subnets of the IPN defined are on it.

### IPN Validation

Multi-Pod relies on an Inter-Pod Network (IPN) which will provide POD-to-POD connectivity. It is crucial to verify that the configuration for the IPN is properly in place. Often faulty or missing configuration is source of unexpected behavior or traffic drop in case of failure scenarios. The configuration for the IPN will be described in detail in this section.

For the next section, reference the following IPN topology:



### Spine to IPN dot1q VLAN-4 sub-interfaces connectivity

Spine to IPN point-to-point connectivity is achieved with sub-interfaces on VLAN-4. The first validation for this connectivity is to test IP reachability between the spines and the IPN devices.

To do so, determine the correct interface and verify it is showing as up.

```
S1P1-Spine201# show ip int brief vrf overlay-1 | grep 172.16.101.2
eth1/29.29      172.16.101.2/30      protocol-up/link-up/admin-up

S1P1-Spine201# show ip interface eth1/29.29
IP Interface Status for VRF "overlay-1"
eth1/29.29, Interface status: protocol-up/link-up/admin-up, ioid: 67, mode: external
IP address: 172.16.101.2, IP subnet: 172.16.101.0/30
IP broadcast address: 255.255.255.255
IP primary address route-preference: 0, tag: 0
```

```

S1P1-Spine201# show system internal ethpm info interface Eth1/29.29
Ethernet1/29.29 - if_index: 0x1A01C01D
Router MAC address: 00:22:bd:f8:19:ff
Admin Config Information:
  state(up), mtu(9150), delay(1), vlan(4), cfg-status(valid)
  medium(broadcast)
Operational (Runtime) Information:
  state(up), mtu(9150), Local IOD(0x43), Global IOD(0x43), vrf(enabled)
  reason(None)
  bd_id(29)
Information from SDB Query (IM call)
  admin state(up), runtime state(up), mtu(9150),
  delay(1), bandwidth(40000000), vlan(4), layer(L3),
  medium(broadcast)
  sub-interface(0x1a01c01d) from parent port(0x1a01c000)/Vlan(4)
Operational Bits:

User config flags: 0x1
  admin_router_mac(1)

Sub-interface FSM state(3)
No errors on sub-interface
Information from GLDB Query:
Router MAC address: 00:22:bd:f8:19:ff

```

After verifying the Interface is up, now test point-to-point IP connectivity:

```

S1P1-Spine201# iping -V overlay-1 172.16.101.1
PING 172.16.101.1 (172.16.101.1) from 172.16.101.2: 56 data bytes
64 bytes from 172.16.101.1: icmp_seq=0 ttl=255 time=0.839 ms
64 bytes from 172.16.101.1: icmp_seq=1 ttl=255 time=0.719 ms
^C
--- 172.16.101.1 ping statistics ---
2 packets transmitted, 2 packets received, 0.00% packet loss
round-trip min/avg/max = 0.719/0.779/0.839 ms
S1P1-Spine201#

```

If there is any connectivity issue, verify cabling and configuration on the remote IPN (IPN1).

```

IPN1# show ip interface brief | grep 172.16.101.1
Eth1/33          172.16.101.101  protocol-up/link-up/admin-up
Eth1/35          172.16.101.105  protocol-up/link-up/admin-up
Eth1/53.4       172.16.101.1    protocol-up/link-up/admin-up

IPN1# show run int Eth1/53.4
interface Ethernet1/53.4
description to spine 1pod1

```

```

mtu 9150
encapsulation dot1q 4
ip address 172.16.101.1/30
ip ospf cost 100
ip ospf network point-to-point
ip router ospf 1 area 0.0.0.0
ip pim sparse-mode
ip dhcp relay address 10.0.0.3
no shutdown
IPN1#

```

### OSPF configuration

OSPF is used as the routing protocol to connect Pod1 and Pod2 together within ACI VRF 'overlay-1'. The following can be referenced as a generic flow to validate if OSPF is coming up between spine and IPN device.

```

S1P1-Spine201# show ip ospf neighbors vrf overlay-1
OSPF Process ID default VRF overlay-1
Total number of neighbors: 2
Neighbor ID    Pri State           Up Time  Address      Interface
172.16.101.201  1 FULL/ -          08:39:35 172.16.101.1 Eth1/29.29
172.16.101.202  1 FULL/ -          08:39:34 172.16.101.9  Eth1/30.30

S1P1-Spine201# show ip ospf interface vrf overlay-1
Ethernet1/29.29 is up, line protocol is up
  IP address 172.16.101.2/30, Process ID default VRF overlay-1, area backbone
  Enabled by interface configuration
  State P2P, Network type P2P, cost 1
  Index 67, Transmit delay 1 sec
  1 Neighbors, flooding to 1, adjacent with 1
  Timer intervals: Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello timer due in 00:00:10
  No authentication
  Number of opaque link LSAs: 0, checksum sum 0
loopback0 is up, line protocol is up
  IP address 10.0.200.66/32, Process ID default VRF overlay-1, area backbone
  Enabled by interface configuration
  State LOOPBACK, Network type LOOPBACK, cost 1
loopback14 is up, line protocol is up
  IP address 172.16.1.4/32, Process ID default VRF overlay-1, area backbone
  Enabled by interface configuration
  State LOOPBACK, Network type LOOPBACK, cost 1
Ethernet1/30.30 is up, line protocol is up
  IP address 172.16.101.10/30, Process ID default VRF overlay-1, area backbone
  Enabled by interface configuration
  State P2P, Network type P2P, cost 1
  Index 68, Transmit delay 1 sec
  1 Neighbors, flooding to 1, adjacent with 1
  Timer intervals: Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello timer due in 00:00:09

```

```
No authentication
  Number of opaque link LSAs: 0, checksum sum 0
```

```
IPN1# show ip ospf neighbors
```

```
OSPF Process ID 1 VRF default
```

```
Total number of neighbors: 5
```

Neighbor ID	Pri	State	Up Time	Address	Interface
172.16.101.203	1	FULL/-	4d12h	172.16.101.102	Eth1/33
172.16.101.202	1	FULL/-	4d12h	172.16.101.106	Eth1/35
172.16.110.201	1	FULL/-	4d12h	172.16.110.2	Eth1/48
<b>172.16.1.4</b>	<b>1</b>	<b>FULL/-</b>	08:43:39	172.16.101.2	Eth1/53.4
<b>172.16.1.6</b>	<b>1</b>	<b>FULL/-</b>	08:43:38	172.16.101.6	Eth1/54.4

When OSPF is up between all spines and IPN devices, all the Pod TEP pools can be seen within the IPN routing tables.

```
IPN1# show ip ospf database 10.0.0.0 detail
```

```
OSPF Router with ID (172.16.101.201) (Process ID 1 VRF default)
```

```
Type-5 AS External Link States
```

```
LS age: 183
```

```
Options: 0x2 (No TOS-capability, No DC)
```

```
LS Type: Type-5 AS-External
```

```
Link State ID: 10.0.0.0 (Network address)
```

```
Advertising Router: 172.16.1.4
```

```
LS Seq Number: 0x80000026
```

```
Checksum: 0x2da0
```

```
Length: 36
```

```
Network Mask: /16
```

```
  Metric Type: 2 (Larger than any link state path)
```

```
  TOS: 0
```

```
  Metric: 20
```

```
  Forward Address: 0.0.0.0
```

```
  External Route Tag: 0
```

```
LS age: 183
```

```
Options: 0x2 (No TOS-capability, No DC)
```

```
LS Type: Type-5 AS-External
```

```
Link State ID: 10.0.0.0 (Network address)
```

```
Advertising Router: 172.16.1.6
```

```
LS Seq Number: 0x80000026
```

```
Checksum: 0x21aa
```

```
Length: 36
```

```
Network Mask: /16
```

```
  Metric Type: 2 (Larger than any link state path)
```

```
  TOS: 0
```

```
  Metric: 20
```

```
  Forward Address: 0.0.0.0
```

```
  External Route Tag: 0
```

```
IPN1# show ip ospf database 10.1.0.0 detail
```

```
OSPF Router with ID (172.16.101.201) (Process ID 1 VRF default)
```

```
Type-5 AS External Link States
```

```

LS age: 1779
Options: 0x2 (No TOS-capability, No DC)
LS Type: Type-5 AS-External
Link State ID: 10.1.0.0 (Network address)
Advertising Router: 172.16.2.4
LS Seq Number: 0x8000022
Checksum: 0x22ad
Length: 36
Network Mask: /16
    Metric Type: 2 (Larger than any link state path)
    TOS: 0
    Metric: 20
    Forward Address: 0.0.0.0
    External Route Tag: 0
LS age: 1780
Options: 0x2 (No TOS-capability, No DC)
LS Type: Type-5 AS-External
Link State ID: 10.1.0.0 (Network address)
Advertising Router: 172.16.2.6
LS Seq Number: 0x8000022
Checksum: 0x16b7
Length: 36
Network Mask: /16
    Metric Type: 2 (Larger than any link state path)
    TOS: 0
    Metric: 20
    Forward Address: 0.0.0.0
    External Route Tag: 0

IPN1# show ip route 10.0.0.0
IP Route Table for VRF "default"
'*' denotes best ucast next-hop
 '**' denotes best mcast next-hop
 '[x/y]' denotes [preference/metric]
 '%<string>' in via output denotes VRF <string>
10.0.0.0/16, ubest/mbest: 2/0
 *via 172.16.101.2, Eth1/53.4, [110/20], 08:39:17, ospf-1, type-2
 *via 172.16.101.6, Eth1/54.4, [110/20], 08:39:17, ospf-1, type-2

IPN1# show ip route 10.1.0.0
IP Route Table for VRF "default"
'*' denotes best ucast next-hop
 '**' denotes best mcast next-hop
 '[x/y]' denotes [preference/metric]
 '%<string>' in via output denotes VRF <string>
10.1.0.0/16, ubest/mbest: 1/0
 *via 172.16.101.102, Eth1/33, [110/20], 08:35:25, ospf-1, type-2

```

Notice on IPN1 for the remote Pod (Pod2), only the most optimal route is shown in the 'show ip route' command.

### DHCP relay configuration

Switch nodes receive their infra TEP address utilizing DHCP towards the APICs. All APICs will typically receive the discover, but it is the first APIC to receive the discover and present an offer which will allocate the TEP address. To account for this in a Multi-Pod scenario, configure DHCP relay on the IPN to receive these discovers and unicast them towards the APICs. Generally, configure all IPN spine-facing interfaces with IP helpers pointing to all APICs. This will futureproof the IPN config if APIC is moved due to recabling, a standby APIC fails over, or any other scenarios that involve an APIC moving to a new Pod.

In this scenario, that means configuring IPN1 Eth1/53.4 and Eth1/54.4 with IP helpers pointing to all APICs:

```
interface Ethernet1/53.4
description to spine 1pod1
mtu 9150
encapsulation dot1q 4
ip address 172.16.101.1/30
ip ospf cost 100
ip ospf network point-to-point
ip router ospf 1 area 0.0.0.0
ip pim sparse-mode
ip dhcp relay address 10.0.0.1
ip dhcp relay address 10.0.0.2
ip dhcp relay address 10.0.0.3
no shutdown

interface Ethernet1/54.4
description to spine 2pod1
mtu 9150
encapsulation dot1q 4
ip address 172.16.101.5/30
ip ospf cost 100
ip ospf network point-to-point
ip router ospf 1 area 0.0.0.0
ip pim sparse-mode
ip dhcp relay address 10.0.0.1
ip dhcp relay address 10.0.0.2
ip dhcp relay address 10.0.0.3
no shutdown
```

From IPN3:

```
interface Ethernet1/53.4
description to spine 1pod2
mtu 9150
encapsulation dot1q 4
ip address 172.16.101.17/30
ip ospf cost 100
ip ospf network point-to-point
ip router ospf 1 area 0.0.0.0
ip pim sparse-mode
ip dhcp relay address 10.0.0.1
ip dhcp relay address 10.0.0.2
ip dhcp relay address 10.0.0.3
no shutdown

interface Ethernet1/54.4
description to spine 2pod2
mtu 9150
encapsulation dot1q 4
ip address 172.16.101.21/30
ip ospf cost 100
ip ospf network point-to-point
ip router ospf 1 area 0.0.0.0
ip pim sparse-mode
ip dhcp relay address 10.0.0.1
ip dhcp relay address 10.0.0.2
ip dhcp relay address 10.0.0.3
no shutdown
```

### MTU

If OSPF is not coming up (EXCHANGE or EXSTART) between spine and IPN device, make sure to validate that MTU matches between devices.

### RP configuration

With PIM BiDir, the Rendezvous Point (RP) is not part of the datapath. For functional multicast, each IPN device need only have a route to the RP address. Redundancy can be achieved using a Phantom RP configuration. In this case, Anycast RP is not a valid redundancy method due to not having a source to exchange via Multicast Source Discovery Protocol (MSDP).

In a Phantom RP design, the RP is a non-existent address in a reachable subnet. In the below config, assume the multicast range configured in the APIC initial setup is the default 225.0.0.0/15. If it was changed in APIC initial setup, IPN configurations must be aligned.

The loopback1 below is the phantom-rp loopback. It must be injected in OSPF; however, it can't be used as OSPF router-id. A separate loopback (loopback0) must be used for that.

IPN1 config:

```
interface loopback1
description IPN1-RP-Loopback
ip address 172.16.101.221/30
ip ospf network point-to-point
ip router ospf 1 area 0.0.0.0
ip pim sparse-mode
ip pim rp-address 172.16.101.222 group-list 225.0.0.0/15 bidir
ip pim rp-address 172.16.101.222 group-list 239.255.255.240/32 bidir
```

IPN2 config:

```
ip pim rp-address 172.16.101.222 group-list 225.0.0.0/15 bidir
ip pim rp-address 172.16.101.222 group-list 239.255.255.240/32 bidir
```

IPN3 config:

```
interface loopback1
description IPN3-RP-Loopback
ip address 172.16.101.221/29
ip ospf network point-to-point
ip router ospf 1 area 0.0.0.0
ip pim sparse-mode
ip pim rp-address 172.16.101.222 group-list 225.0.0.0/15 bidir
ip pim rp-address 172.16.101.222 group-list 239.255.255.240/32 bidir
```

IPN4 config:

```
ip pim rp-address 172.16.101.222 group-list 225.0.0.0/15 bidir
ip pim rp-address 172.16.101.222 group-list 239.255.255.240/32 bidir
```

The subnet mask on the loopback cannot be a /32. To use IPN1 as the primary device in the Phantom RP design, use a /30 subnet mask to take advantage of the most specific

route being preferred in the OSPF topology. IPN3 will be the secondary device in the Phantom RP design, so use a /29 subnet mask to make it a less specific route. The /29 will only get used if something happens to stop the /30 from existing and subsequently existing within the OSPF topology.

### Troubleshooting the 1st Remote Pod spine joining the fabric

The following steps outlines the process that the 1st Remote Pod Spine takes to join the fabric:

- 1 The spine will do DHCP on its sub-interface facing the IPN. The DHCP Relay config will carry this discover to the APICs. The APICs will respond if the spine was added in the Fabric Membership. The IP address that gets offered is the IP address configured on the Multi-Pod L3Out.
- 2 The spine will install a route towards the DHCP server that offered the IP address as a static route towards the other end of the point-to-point interface.
- 3 The spine will download a bootstrap file from the APIC through the static route.
- 4 The spine will get configured based on the bootstrap file to bring up VTEP, OSPF and BGP to join the fabric.

From the APIC, validate if the L3Out IP is properly configured to be offered: (our Spine 401 has serial FDO22472FCV)

```
bdso1-aci37-apic1# moquery -c dhcpExtIf
# dhcp.ExtIf
ifId      : eth1/30
childAction :
dn        : client-[FDO22472FCV]/if-[eth1/30]
ip        : 172.16.101.26/30
lcOwn     : local
modTs     : 2019-10-01T09:51:29.966+00:00
name      :
nameAlias :
relayIp   : 0.0.0.0
rn        : if-[eth1/30]
status    :
subIfId   : unspecified
# dhcp.ExtIf
ifId      : eth1/29
childAction :
```

```

dn      : client-[FD022472FCV]/if-[eth1/29]
ip      : 172.16.101.18/30
lcOwn  : local
modTs   : 2019-10-01T09:51:29.966+00:00
name    :
nameAlias :
relayIp : 0.0.0.0
rn      : if-[eth1/29]
status  :
subIfId : unspecified

```

Validate if the IPN-facing interface received the expected IP address matching L3Out configuration done in infra Tenant.

```

S1P2-Spine401# show ip interface brief | grep eth1/29
eth1/29          unassigned          protocol-up/link-up/admin-up
eth1/29.29      172.16.101.18/30  protocol-up/link-up/admin-up
S1P2-Spine401#

```

Now IP connectivity has been established from the spine to the APIC and connectivity through ping can be verified:

```

S1P2-Spine401# iping -V overlay-1 10.0.0.1
PING 10.0.0.1 (10.0.0.1) from 172.16.101.18: 56 data bytes
64 bytes from 10.0.0.1: icmp_seq=0 ttl=60 time=0.345 ms
64 bytes from 10.0.0.1: icmp_seq=1 ttl=60 time=0.294 ms
^C
--- 10.0.0.1 ping statistics ---
2 packets transmitted, 2 packets received, 0.00% packet loss
round-trip min/avg/max = 0.294/0.319/0.345 ms
S1P2-Spine401#

```

The spine will now bring up the OSPF to the IPN and setup a loopback for the router id:

```

S1P2-Spine401# show ip ospf neighbors vrf overlay-1
OSPF Process ID default VRF overlay-1
Total number of neighbors: 2
Neighbor ID   Pri State           Up Time  Address          Interface
172.16.101.204 1 FULL/-          00:04:16 172.16.101.25   Eth1/30.30
172.16.101.203 1 FULL/-          00:04:16 172.16.101.17   Eth1/29.29
S1P2-Spine401# show ip ospf interface vrf overlay-1
loopback8 is up, line protocol is up
  IP address 172.16.2.4/32, Process ID default VRF overlay-1, area backbone
  Enabled by interface configuration
  State LOOPBACK, Network type LOOPBACK, cost 1

```

## 58 Fabric discovery

```
Ethernet1/30.30 is up, line protocol is up
  IP address 172.16.101.26/30, Process ID default VRF overlay-1, area backbone
  Enabled by interface configuration
  State P2P, Network type P2P, cost 1
  Index 68, Transmit delay 1 sec
  1 Neighbors, flooding to 1, adjacent with 1
  Timer intervals: Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello timer due in 00:00:07
  No authentication
  Number of opaque link LSAs: 0, checksum sum 0
Ethernet1/29.29 is up, line protocol is up
  IP address 172.16.101.18/30, Process ID default VRF overlay-1, area backbone
  Enabled by interface configuration
  State P2P, Network type P2P, cost 1
  Index 67, Transmit delay 1 sec
  1 Neighbors, flooding to 1, adjacent with 1
  Timer intervals: Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello timer due in 00:00:04
  No authentication
  Number of opaque link LSAs: 0, checksum sum 0
```

The spine will now receive its PTEP through DHCP:

```
S1P2-Spine401# show ip interface vrf overlay-1 | egrep -A 1 status
lo0, Interface status: protocol-up/link-up/admin-up, iod: 4, mode: ptep
IP address: 10.1.88.67, IP subnet: 10.1.88.67/32
```

The spine will move from Discovering to Active and is fully discovered:

```
bdsol-aci37-apic1# acidiag fmvread
```

ID	Pod ID	Name	Serial Number	IP Address	Role	State	LastUpdMsgId
101	1	S1P1-Leaf101	FD0224702JA	10.0.160.64/32	leaf	active	0
102	1	S1P1-Leaf102	FD0223007G7	10.0.160.67/32	leaf	active	0
201	1	S1P1-Spine201	FD022491705	10.0.160.65/32	spine	active	0
202	1	S1P1-Spine202	FD0224926Q9	10.0.160.66/32	spine	active	0
<b>401</b>	<b>2</b>	<b>S1P2-Spine401</b>	<b>FD022472FCV</b>	<b>10.1.88.67/32</b>	<b>spine</b>	<b>active</b>	<b>0</b>

Please do know that we can only discover a remote spine when it has at least 1 leaf switch connected to it.

## Verify remaining leaf and spine switches

The rest of the Pod is now discovered as per the normal Pod bring up procedure, as discussed in the section "Initial fabric setup".

## Check remote Pod APIC

To discover the 3rd APIC, the following process is followed:

- The leaf301 creates a static route to the directly connected APIC (APIC3) based on LLDP (same as single Pod case). The remote APIC will receive an IP address out of the POD1 IP Pool. We will create this route as a /32.
- Leaf301 advertises this route using IS-IS to Spine401 and Spine402 (same as single Pod case)
- Spine401 and Spine402 redistribute this route into OSPF towards IPN
- Spine201 and Spine202 redistribute this route from OSPF to IS-IS in Pod1
- Now connectivity is established between APIC3 and APIC1 and APIC2
- APIC3 can now join the cluster

In order to confirm, use the following checks:

The Leaf301 creates a static route to the directly connected APIC (APIC3) based on LLDP (same as Single Pod case)

```
S1P2-Leaf301# show ip route 10.0.0.3 vrf overlay-1
IP Route Table for VRF "overlay-1"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
10.0.0.3/32, ubest/mbest: 2/0
 *via 10.1.88.64, eth1/50.14, [115/12], 00:07:21, isis-isis_infra, isis-l1-ext
 *via 10.1.88.67, eth1/49.13, [115/12], 00:07:15, isis-isis_infra, isis-l1-ext
 via 10.0.0.3, vlan9, [225/0], 07:31:04, static
```

Leaf301 advertises this route using IS-IS to Spine401 and Spine402 (same as single Pod case)

Spine401 and Spine402 leak this route into OSPF towards IPN

```
S1P2-Spine401# show ip route 10.0.0.3 vrf overlay-1
IP Route Table for VRF "overlay-1"
'*' denotes best ucast next-hop
***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
10.0.0.3/32, ubest/mbest: 1/0
    *via 10.1.88.65, eth1/2.35, [115/11], 00:17:38, isis-isis_infra, isis-l1-ext S1P2-Spine401#

IPN3# show ip route 10.0.0.3
IP Route Table for VRF "default"
'*' denotes best ucast next-hop
***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
10.0.0.3/32, ubest/mbest: 2/0
    *via 172.16.101.18, Eth1/53.4, [110/20], 00:08:05, ospf-1, type-2
    *via 172.16.101.22, Eth1/54.4, [110/20], 00:08:05, ospf-1, type-2

S1P1-Spine201# show ip route vrf overlay-1 10.0.0.3
IP Route Table for VRF "overlay-1"
'*' denotes best ucast next-hop
***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
10.0.0.3/32, ubest/mbest: 2/0
    *via 172.16.101.1, eth1/29.29, [110/20], 00:08:59, ospf-default, type-2
    *via 172.16.101.9, eth1/30.30, [110/20], 00:08:59, ospf-default, type-2
    via 10.0.160.64, eth1/1.36, [115/12], 00:18:19, isis-isis_infra, isis-l1-ext
    via 10.0.160.67, eth1/2.35, [115/12], 00:18:19, isis-isis_infra, isis-l1-ext
S1P1-Spine201#
```

Now connectivity is established between APIC3 and APIC1 and APIC2

APIC3 can now join the cluster

```
apic1# show controller
Fabric Name       : POD37
Operational Size  : 3
Cluster Size     : 3
Time Difference   : 133
Fabric Security Mode : PERMISSIVE
```

ID	Pod	Address	In-Band IPv4	In-Band IPv6	00B IPv4	00B IPv6
Version		Flags	Serial Number	Health		
1*	1	10.0.0.1	0.0.0.0	fc00::1	10.48.176.57	
		fe80::d6c9:3cff:fe51:cb82	4.2(1i)	crva-	WZP22450H82	fully-fit
2	1	10.0.0.2	0.0.0.0	fc00::1	10.48.176.58	
		fe80::d6c9:3cff:fe51:ae22	4.2(1i)	crva-	WZP22441A22	fully-fit
<b>3</b>	<b>2</b>	<b>10.0.0.3</b>	<b>0.0.0.0</b>	<b>fc00::1</b>	<b>10.48.176.59</b>	
		<b>fe80::d6c9:3cff:fe51:a30a</b>	<b>4.2(1i)</b>	<b>crva-</b>	<b>WZP22441B0T</b>	<b>fully-fit</b>

Flags - c:Commissioned | r:Registered | v:Valid Certificate | a:Approved | f/s:Failover fail/success  
 (\*)Current (~)Standby (+)AS  
 apic1#

Ping from APIC1 to a remote device in Pod2 to validate connectivity via the following ping: (make sure to source from the local interface, in APIC1 case 10.0.0.1)

```
apic1# ping 10.0.0.3 -I 10.0.0.1
PING 10.0.0.3 (10.0.0.3) from 10.0.0.1 : 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=58 time=0.132 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=58 time=0.236 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=58 time=0.183 ms
^C
--- 10.0.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2048ms
rtt min/avg/max/mdev = 0.132/0.183/0.236/0.045 ms
apic1#
```

## Scenarios

### Spine cannot ping the IPN

This is most likely caused by:

- A misconfiguration in the ACI Access Policies.
- A misconfiguration in the IPN configuration.

Please refer to the "Troubleshooting workflow" in this chapter and review:

- Verify ACI Policies.
- IPN Validation.

### Remote spine is not joining fabric

This is most likely caused by:

- DHCP relay issue on IPN network.
- Spine-to-APIC IP reachability over the IPN network.

Please refer to the "Troubleshooting workflow" in this chapter and review:

- Verify ACI Policies.
- IPN Validation.
- Troubleshoot 1st fabric join.

Make sure to validate that there is at least 1 leaf connected to the remote spine and that the spine has an LLDP adjacency with this leaf.

### APIC in Pod2 is not joining fabric

This is typically caused by a mistake in the APIC initial setup dialog assuming the remote Pod leaf and spine switches were able to correctly join the fabric. In a correct setup, expect the following 'avread' output (working APIC3 join scenario):

```

apic1# avread
Cluster:
-----
fabricDomainName    POD37
discoveryMode       PERMISSIVE
clusterSize         3
version             4.2(1i)
drrMode             OFF
operSize            3
APICs:
-----

```

	APIC 1	APIC 2	APIC 3
version	4.2(1i)	4.2(1i)	4.2(1i)
address	10.0.0.1	10.0.0.2	10.0.0.3
oobAddress	10.48.176.57/24	10.48.176.58/24	10.48.176.59/24
routableAddress	0.0.0.0	0.0.0.0	0.0.0.0
tepAddress	10.0.0.0/16	10.0.0.0/16	<b>10.0.0.0/16</b>

podId	1	1	2
chassisId	7e34872e-.d3052cda	84debc98-.e207df70	89b73e48-.f6948b98
cntrlSbst_serial	(APPROVED,WZP22450H82)	(APPROVED,WZP22441A22)	(APPROVED,WZP22441B0T)
active	YES	YES	YES
flags	cra-	cra-	cra-
health	255	255	255
apic1#			

Notice that APIC3 (in the remote Pod) is configured with podId 2 and the tepAddress of Pod1.

Verify the original APIC3 setup settings by using the following command:

```
apic3# cat /data/data_admin/sam_exported.config
Setup for Active and Standby APIC
fabricDomain = POD37
fabricID = 1
systemName = bdsol-aci37-apic3
controllerID = 3
tepPool = 10.0.0/16
infraVlan = 3937
clusterSize = 3
standbyApic = NO
enableIPv4 = Y
enableIPv6 = N
firmwareVersion = 4.2(1i)
ifcIpAddr = 10.0.0.3
apicX = NO
podId = 2
oobIpAddr = 10.48.176.59/24
```

If a mistake occurs, login to APIC3 and execute 'acidiag touch setup' and 'acidiag reboot'.

### POD-to-POD BUM traffic not working

This is most likely caused by:

- The lack of an RP in the IP network
- The RP not reachable by the ACI fabricGeneral Multicast misconfiguration on the IPN devices

Please refer to the "Troubleshooting workflow" in this chapter and review:

- IPN Validation

Also make sure one of the IPN RP devices is online.

#### **After 1 IPN device failed, BUM traffic is being dropped**

As described in the IPN Validation in the troubleshooting workflow, use a Phantom RP to guarantee when the primary RP goes down that a secondary RP is available. Make sure to review the "IPN Validation" section and verify the correct validation.

#### **Inter-Pod endpoint connectivity is broken within the same EPG**

This is most likely caused by a misconfiguration in the Multi-Pod setup, make sure to validate the troubleshooting workflow and verify the entire flow. If this looks OK, please refer to the "Multi-Pod forwarding" section in the chapter "Intra-Fabric forwarding" to further troubleshoot this issue.

# Device replacement

## Introduction

During the evolution of an ACI fabric, it will become necessary to replace various components including: APICs, leaf switches, spine switches, and IPN devices. The most common reasons for replacement include RMAs and hardware upgrades. These procedures are well documented in the Cisco Install/Upgrade guides and the most recent guide should be read prior to replacement. This section will include a deeper look into how the procedures work under the hood; as well as walk through several of the most common troubleshooting scenarios.

## Procedures and verification

### Hardware replacement

#### Leaf

A leaf from the RMA depot will arrive running NXOS software. Please reference to the below section called 'Problem: Arrives in NXOS mode' to properly convert the leaf to ACI mode. If using a leaf from a different fabric or with previous configuration, make sure to use the commands 'acidiag touch clean' and 'reload'.

After the above steps are completed and the new leaf switch is ready for registration, remove the leaf to be replaced from the fabric via the 'Remove from Controller' option.

The 'Remove from Controller' option will completely remove the node from the APIC, releasing the node ID, SN association, and TEP address which was assigned by the APIC. These processes are desired when replacing a switch node. The 'Decommission' option is only used when the expectation is that the same node will rejoin the fabric with the same node ID and SN.

When the leaf switch to be replaced is no longer seen on the **Fabric Membership** page, the new leaf can be connected to the fabric via the spine interfaces. Once the leaf is discovered by the APIC, it will show up in the Fabric Inventory and be ready for

registration. If the device to be replaced has not yet released its node ID, and a new switch is registered with the same node ID, a fault will be thrown referencing the fact that the ID is already associated to another leaf node. The fault should clear after some time. If the new node does not show up on the Fabric Membership submenu, there could be a cabling issue; this can be verified by viewing the LLDP neighbors via the 'show lldp neighbors detail' command on the spine switches connecting to the newly attached leaf switch. For more detail on the Fabric Discovery process, please reference the "Initial fabric setup" chapter.

If the infra VLAN is modified, all leaf nodes must be clean rebooted at the same time. If all leaf switches are not cleaned at the same time, a clean reloaded switch will come online and receive the old infra VLAN via LLDP from a not-yet-cleaned leaf, and the clean reloaded leaf will fail to register with the APIC. See the "Initial fabric setup" chapter for more details.

Due to platform limitations, VPC pairs cannot be a mix of Gen1 and Gen2 or higher leaf switches. However, at the time of writing, any Gen2 leaf and higher can mix with any other Gen2 leaf or higher.

### Spine

Like a leaf, depending on the HW of the spine (such as modular spine) it could arrive in NXOS mode. Use the procedure "Problem: Arrives in NXOS mode" under the scenarios to perform the conversion.

When replacing a spine switch, the user must consider the **BGP Route Reflector** functionality. As a best practice there must be at least two spine switches configured as BGP Route Reflectors for a Layer 3 Cisco ACI fabric. This configuration is located at 'System > System Settings > BGP Route Reflectors' under Route Reflector Nodes. When replacing or removing a spine switch, ensure the appropriate configuration changes are made to maintain one active Route Reflector, and ensure at least two active Route Reflectors after the changes are completed.

Refer to section "Pod Policies — BGP RR / Date&Time / SNMP" in chapter "Management and core services" for more information on the BGP Route Reflectors.

## APIC

The most important consideration when performing an APIC replacement is the health of the existing APIC cluster. Prior to the replacement, all APICs in the cluster should be reported as Fully Fit. In 4.2, an additional tool was introduced to verify the health of the APIC cluster via CLI:

```

apic1# acdiag cluster
Admin password:
Product-name = APIC-SERVER-L2
Serial-number = FCH2206W0RK
Running...
Checking Core Generation: OK
Checking Wiring and UUID: OK
Checking AD Processes: Running
Checking All Apics in Commission State: OK
Checking All Apics in Active State: OK
Checking Fabric Nodes: OK
Checking Apic Fully-Fit: OK
Checking Shard Convergence: OK
Checking Leadership Degration: Optimal leader for all shards
Ping OOB IPs:
APIC-1: 192.168.4.20 - OK
Ping Infra IPs:
APIC-1: 10.0.0.1 - OK
Checking APIC Versions: Same (4.2(1i))
Checking SSL: OK

Done!

```

When replacing an APIC, make sure to note the initial setup variables of the APIC to be replaced, before performing a decommission of the APIC.

```

apic1# cat /data/data_admin/sam_exported.config
Setup for Active and Standby APIC
fabricDomain = POD37
fabricID = 1
systemName = apic1
controllerID = 1
tepPool = 10.0.0.0/16
infraVlan = 3937
GIPO = 225.0.0.0/15
clusterSize = 3
standbyApic = NO
enableIPv4 = Y
enableIPv6 = N
firmwareVersion = 4.2(1i)
ifcIpAddr = 10.0.0.1

```

```
apicX = N0  
podId = 1  
oobIpAddr = 10.48.176.57/24
```

Prepare the new APIC with the correct software version and re-enter the initial setup values referenced earlier. When the initial setup is complete and the APIC is fully booted, recommission it to the fabric from UI of one of the other APICs in the cluster.

### IPN device replacement

In a Multi-Pod environment, it might be necessary to replace one of the devices being used for the IPN (Inter-Pod Network). Prior to the replacement, the IPN network must have **PIM Bidirectional Rendezvous Point Redundancy** configured in the form of **Phantom RPs**. Without Phantom RPs in place, if the node replaced was the RP, there would be a PIM convergence and packet loss would be seen for all BUM traffic sent across the IPN.

Please refer to "RP configuration" in "Multi-Pod Discovery" chapter for more information on how to configure Phantom RP.

### Clean reload of APIC/leaf/spine

In certain scenarios, the best option for recovering a leaf/spine that won't join the fabric is to perform a clean reload of the device.

**It is not recommended to perform a clean reload on a device that is waiting for its turn to upgrade. Clean reload of any device can take an extended period of time.**

The 'acidia touch' command has two options, clean and setup. The **clean** option removes all policy data while retaining the APIC network configuration (such as fabric name, IP address, login). The **setup** option removes both policy data and the APIC network configuration. The setup option is most commonly used when moving devices across Pods, as the Pod ID must be changed, and normally the management network will need to update as well.

## APIC

```
fab1-apic1# acdiag touch clean
This command will wipe out this device, Proceed? [y/N] y
fab1-apic1# acdiag reboot
This command will restart this device, Proceed? [y/N] y
```

## Leaf/Spine

```
fab1-leaf101# acdiag touch clean
This command will wipe out this device, Proceed? [y/N] y
fab1-leaf101# reload
This command will reload the chassis, Proceed (y/n)? [n]: y
```

The 'acdiag touch clean' command works by putting a hidden file on the leaf in /mnt/pss called .clean. When the leaf is booted, a shell script runs that checks to see if .clean file is present. In the event that .clean file exists under /mnt/pss, policy configuration is wiped and configuration is redownloaded from the APIC. If this command is entered and the node is not reloaded, then the file will still be present and the policy will still be wiped upon the next reload, no matter how much time has elapsed since the touch clean was entered.

## Troubleshooting scenarios

### Problem: Arrives in NXOS mode

#### Verification

Sometimes when a switch is shipped via RMA, it can arrive with NXOS software that has not yet been configured via the Power On Auto Provisioning (POAP) process. When the user consoles into this device they will see some form of the following message:

```
Abort Auto Provisioning and continue with normal setup?(yes/no)
```

If the device has already gone through POAP, the simplest way to determine if a leaf is running standalone NXOS code is to look for the 'NXOS image file' line in the 'show version' output. If such output is present, the leaf is running standalone code and will

need to be converted to ACI mode. The presence of Kickstart and system images can be verified and will only be present on a leaf running an ACI image, by looking at the image itself, which will be n9000 on standalone and aci-n9000 on ACI.

### Standalone NXOS

```
nxos-n9k# show version
Cisco Nexus Operating System (NX-OS) Software
.
.
.
Software
  BIOS: version 07.17
  NXOS: version 6.1(2)I3(4)
  BIOS compile time: 09/10/2014
  NXOS image file is: bootflash:///n9000-dk9.6.1.2.I3.4.bin
  NXOS compile time: 3/18/2015 0:00:00 [03/18/2015 07:49:10]
```

### ACI

```
aci-leaf101# show version
Cisco Nexus Operating System (NX-OS) Software
.
.
.
Software
  BIOS: version 07.66
  kickstart: version 14.2(1i) [build 14.2(1i)]
  system: version 14.2(1i) [build 14.2(1i)]
  PE: version 4.2(1i)
  BIOS compile time: 06/11/2019
  kickstart image file is: /bootflash/aci-n9000-dk9.14.2.1i.bin
  kickstart compile time: 09/07/2019 10:25:16 [09/07/2019 10:25:16]
  system image file is: /bootflash/auto-s
  system compile time: 09/07/2019 10:25:16 [09/07/2019 10:25:16]
```

### Solution

If the switch was shipped running NXOS code, it will need to be converted to ACI mode. The switch should be shipped with both the NXOS and the ACI image in the bootflash, although this is not always the case. The ACI image will start with 'aci-n9000'. If the ACI image is not present, then it will need to be manually loaded onto the bootflash. This can be performed via the USB connection (local access needed) or via SCP from the

APIC directly (assuming both devices are connected via a management network). Here are the instructions to copy the image via SCP:

- 1 `nexus-9000(config)# feature scp-server`
- 2 `apic1# scp -r /firmware/fwrepos/fwrepo/switch-image-name admin@standalone_switch:switch-image-name`

The leaf will then need to be configured to not boot the NXOS image, save the configuration, change the boot statements to ACI.

- 1 `(config)# no boot nxos`
- 2 `(config)# copy run start`
- 3 `(config)# boot aci bootflash:<aci-image-name>`
- 4 `(config)# reload`

### Problem: Leaf/Spine EPLD/FPGA not correct, F1582

#### Verification

The following faults will be seen in the Faults for the Nexus 9000 ACI switch.

```
F1582 FPGA version mismatch detected. Running version:0x(z) Expected version:0x(y)
```

From the APIC CLI, search for all instances of Fault F1582:

```
moquery -c faultInst -f 'fault.Inst.code=="F1582"'
```

### EPLD notes

The Cisco Nexus 9000 Series ACI-mode switches contain several programmable logical devices (PLDs) that provide hardware functionalities in all modules. Cisco provides electronic programmable logic device (EPLD) image upgrades to enhance hardware functionality or to resolve known issues. PLDs include electronic programmable logic devices (EPLDs), field programmable gate arrays (FPGAs), and complex programmable logic devices (CPLDs), but they do not include ASICs.

The term EPLD is used to cover both FPGA and CPLDs.

The advantage of having EPLDs for some module functions is that when those functions need to be upgraded, just upgrade their software images instead of replacing their hardware.

EPLD image upgrades for an I/O module disrupt the traffic going through the module because the module must power down briefly during the upgrade. In a modular chassis, the system performs EPLD upgrades on one module at a time, so at any one time the upgrade disrupts only the traffic going through one module.

Cisco provides the latest EPLD images with each release. Typically, these images are the same as provided in earlier releases but occasionally some of these images are updated. These EPLD image updates are not mandatory unless otherwise specified. When Cisco makes an EPLD image upgrade available, these release notes announce their availability, and they can be downloaded from the Cisco web site.

When new EPLD images are available, the upgrades are always recommended if the network environment allows for a maintenance period in which some level of traffic disruption is acceptable. In general, EPLD upgrades will be needed when new hardware functionality is added as a result of a software upgrade.

There may also be various reasons for the need to upgrade EPLD firmware while already in ACI Mode:

- 1 EPLD versions required an upgrade prior to a Cisco NX-OS to ACI Boot Mode conversion and the FPGA/EPLDs were NOT upgraded.
- 2 Leaf/Spine was upgraded manually (instead of a policy upgrade from the APIC), which does not include an EPLD upgrade.

Once the leaf or spine is added to the fabric, then the EPLD will be automatically upgraded with any policy upgrade (normal upgrade initiated from the APIC firmware tab) where a new version of EPLD is available.

### Solution

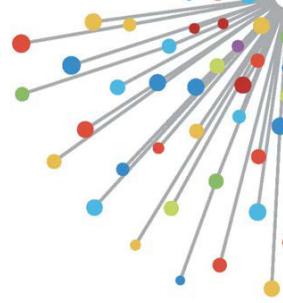
In older versions of ACI, it was necessary to downgrade and then upgrade the leaf/spine in question, but as of 11.2(1m), there are two shell scripts available to the admin user which greatly simplify the process.

```
fab1-leaf101# /bin/check-fpga.sh FpGaDwnGrAdE
```

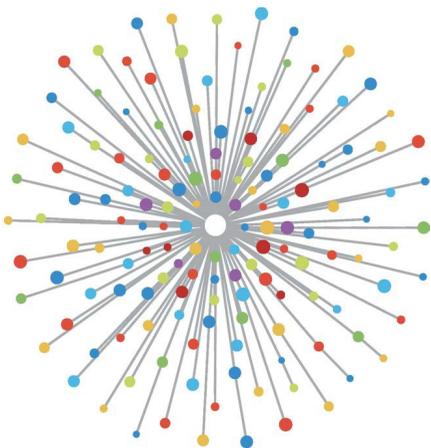
```
fab1-leaf101# /usr/sbin/chassis-power-cycle.sh
```

The '/usr/sbin/chassis-power-cycle.sh' script hard resets power, as compared to a 'reload' which is simply a software restart. When upgrading EPLD, the power needs to be removed entirely to reprogram the firmware on the line cards. If '/usr/sbin/chassis-power-cycle.sh' is not available or does not work, power cables need to be removed for at least 30 seconds and then re-attached to restore power.





# Management and core services



## Overview

When a Cisco ACI fabric is discovered, a number of core and management services should be configured to enable the fabric to be fully operational.

Those services include configuration for:

- Network connectivity of the nodes and controllers.
- BGP Route Reflectors.
- Date and Time Policy (NTP).
- SNMP.

The core and management services should be typically defined as a 'Day Zero' task. The configuration of those services is well described in the "Cisco APIC Basic Configuration Guide" available on Cisco.com.

This chapter will focus on providing the reader with a number of tools and troubleshooting scenarios to verify that those services are configured successfully and functioning as expected.

It is important to understand that failing to properly configure those basic services could affect the operation of the fabric, and sometimes possibly even break specific functionality. For example, if BGP Route Reflectors are not defined on a running ACI fabric, MP-BGP will not be initialized, and as a consequence, L3Out learned routes would not be redistributed in the fabric. This configuration break will possibly prevent tenant connectivity.

Also, failing to configure SNMP and NTP will impact the operation of the fabric as an ACI administrator may not be able to correlate log dates or receive proper alerting for faults.

Finally, failing to properly configure OOB or in-band connectivity of all the nodes and controllers of a fabric would make it more difficult to connect to those devices if troubleshooting becomes necessary.

## In-band and out-of-band management

ACI fabric nodes have two options for management connectivity; out-of-band (OOB), which governs the dedicated physical management port on the back of the device, or in-band (INB), which is provisioned using a specific EPG/BD/VRF in the management tenant with a degree of configurable parameters. There's an OOB EPG present in the management ('mgmt') tenant, but it's there by default and can't be modified. It only allows configuration of Provided OOB Contracts.

On the APIC, the OOB interface is observed in the 'ifconfig' command output as 'oobmgmt' and the in-band interface will be represented by the 'bond.x' interface, where <x> is the encap VLAN configured for the in-band EPG.

```

apic1# ifconfig oobmgmt
oobmgmt: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.4.20 netmask 255.255.255.0 broadcast 192.168.4.255
    inet6 fe80::7269:5aff:feca:2986 prefixlen 64 scopeid 0x20<link>
    ether 70:69:5a:ca:29:86 txqueuelen 1000 (Ethernet)
    RX packets 495815 bytes 852703636 (813.2 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 432927 bytes 110333594 (105.2 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

apic1# ifconfig bond0.300
bond0.300: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1496
    inet 10.30.30.254 netmask 255.255.255.0 broadcast 10.30.30.255
    inet6 fe80::25d:73ff:fec1:8d9e prefixlen 64 scopeid 0x20<link>
    ether 00:5d:73:c1:8d:9e txqueuelen 1000 (Ethernet)
    RX packets 545 bytes 25298 (24.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6996 bytes 535314 (522.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

On the leaf, the OOB interface is seen as 'eth0' in the 'ifconfig' command output and the INB is seen as a dedicated SVI. The user can view the interface with 'ifconfig' or with 'show ip interface vrf mgmt:<vrf>' where <vrf> is the name selected for the in-band VRF.

```
leaf101# show interface mgmt 0
mgmt0 is up
admin state is up,
Hardware: GigabitEthernet, address: 00fc.baa8.2760 (bia 00fc.baa8.2760)
Internet Address is 192.168.4.23/24
MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec
reliability 255/255, txload 1/255, rxload 1/255
Encapsulation ARPA, medium is broadcast
Port mode is routed
full-duplex, 1000 Mb/s
Beacon is turned off
Auto-Negotiation is turned on
Input flow-control is off, output flow-control is off
Auto-mdix is turned off
EtherType is 0x0000
30 seconds input rate 3664 bits/sec, 4 packets/sec
30 seconds output rate 4192 bits/sec, 4 packets/sec
Rx
 14114 input packets 8580 unicast packets 5058 multicast packets
 476 broadcast packets 2494768 bytes
Tx
 9701 output packets 9686 unicast packets 8 multicast packets
 7 broadcast packets 1648081 bytes
```

```
leaf101# show ip interface vrf mgmt:inb
IP Interface Status for VRF "mgmt:inb-vrf"
vlan16, Interface status: protocol-up/link-up/admin-up, iod: 4, mode: pervasive
IP address: 10.30.30.1, IP subnet: 10.30.30.0/24
secondary IP address: 10.30.30.3, IP subnet: 10.30.30.0/24
IP broadcast address: 255.255.255.255
IP primary address route-preference: 0, tag: 0
```

The 'show ip interface vrf mgmt:<vrf>' will show the in-band management BD subnet IP as the secondary IP address; this is expected output.

On spine switches the in-band management IP address is added as a dedicated loopback interface in the 'mgmt:<vrf>' VRF. This implementation is thus different from the in-band management IP implementation on leaf switches.

Observe the 'show ip int vrf mgmt:<vrf>' command output below on a spine switch

```
spine201# show ip interface vrf mgmt:inb
IP Interface Status for VRF "mgmt:inb"
  lo10, Interface status: protocol-up/link-up/admin-up, iof: 98, mode: pervasive
    IP address: 10.30.30.12, IP subnet: 10.30.30.12/32
    IP broadcast address: 255.255.255.255
    IP primary address route-preference: 0, tag: 0
```

Under the System Settings, there is a setting to select either the in-band or out-of-band connectivity preference for the APICs.

Only the traffic sent from the APIC will use the management preference selected in the 'APIC Connectivity Preferences'. The APIC can still receive traffic on either in-band or out-of-band, assuming either is configured. APIC uses the following forwarding logic:

- Packets that come in an interface and go out that same interface.
- Packets sourced from the APIC, destined to a directly connected network, go out the directly connected interface.
- Packets sourced from the APIC, destined to a remote network, prefer in-band or out-of-band based on the APIC Connectivity Preferences.

## APIC Connectivity Preferences

The screenshot shows the APIC System Settings page. The 'System Settings' menu is open, and 'APIC Connectivity Preferences' is selected. The 'APIC Connectivity Preferences' page is displayed, showing the 'Policy' tab. Under 'Properties', the 'Interface to use for external connections:' is set to 'ooband'.

APIC routing table with OOB selected. Observe the metric value of 16 for the oobmgmt interface which is lower than the bond0.300 in-band management interface metric of 32. Meaning the oobmgmt out-of-band management interface will be used for outgoing management traffic.

```

apic1# bash
admin@apic1:~> route -n
Kernel IP routing table
Destination      Gateway         Genmask        Flags Metric Ref    Use Iface
0.0.0.0          192.168.4.1    0.0.0.0        UG    16    0      0 oobmgmt
0.0.0.0          10.30.30.1     0.0.0.0        UG    32    0      0 bond0.300

```

APIC routing table with in-band selected. Observe the bond0.300 in-band management interface's metric is 8 which is now lower than the oobmgmt interface metric of 16. Meaning the bond0.300 in-band management interface will be used for outgoing management traffic.

```
admin@apic1:~> route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 10.30.30.1 0.0.0.0 UG 8 0 0 bond0.300
0.0.0.0 192.168.4.1 0.0.0.0 UG 16 0 0 oobmgmt
```

The leaf and spine node management preferences are not affected by this setting. These connectivity preferences are selected under the protocol policies. Below is an example for NTP.

## Spine or leaf management preferences for NTP

The screenshot displays the APIC interface for configuring NTP. The left sidebar shows the navigation tree with 'Fabric' and 'Fabric Policies' highlighted. The main panel shows the configuration for 'Providers - NTP Server 10.48.37.151'. The 'Management EPG' dropdown is open, showing options: 'default (Out-of-Band)', 'mgmt/default', 'inb\_mgmt (In-Band)', and 'mgmt/default'. The 'Preferred' checkbox is checked, and the 'Minimum Polling Interval' is set to 4 and 'Maximum Polling Interval' is set to 6.

If in-band is selected under the APIC Connectivity Preferences, but then out-of-band is selected under the protocol, which interface with the protocol packet use?

- The APIC Connectivity Preference will always take precedence over the protocol selection on the APIC.
- The leaf nodes are the opposite, they only reference the selection under the protocol.

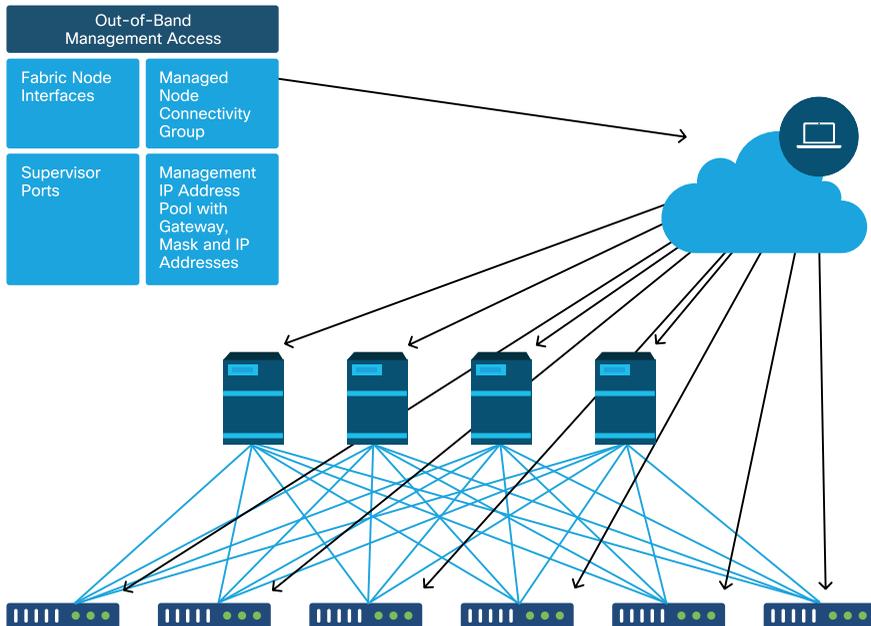
## Scenario: Unable to reach management network

If the user is unable to reach the management network, it may be due to a number of different issues, but they can always use the same methodology to isolate the issue. The assumption in this scenario is that the user cannot reach any devices in the management network from behind their L3Out.

- Verify the APIC connectivity preference. This is outlined in figure 'APIC Connectivity Preferences', and the options are OOB or in-band.
- Depending on which preference is selected, verify the configuration is correct, the interfaces are up, the default gateway is reachable via the selected interface, and there are no drops on the path of the packet.

Do not forget to check for faults in each section of configuration in the GUI. However, some configuration mistakes can manifest in unexpected states, but a fault may be generated in another section than the one the user would initially consider.

## Out-of-Band Management Access



### Out-of-band configuration verification

For out-of-band configuration, there are four folders to verify under a special tenant called 'mgmt':

- Node Management Addresses.
- Node Management EPGs.
- Out-of-band Contracts (under Contracts).
- External Network Instance Profiles.

Node Management Addresses can either be assigned statically or from a pool. Below is an example of static address assignment. Verify that the type out-of-band IP addresses are assigned and that the default gateway is correct.

### Static Node Management Addresses GUI verification

The screenshot displays the Cisco APIC interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Tenants' tab is active, and the 'mgmt' tenant is selected. The left sidebar shows a tree view with 'mgmt' expanded, and 'Static Node Management Addresses' highlighted. The main content area shows a table of Static Node Management Addresses.

Node ID	Name	Type	EPG	IPv4 Address	IPv4 Gateway	IPv6 Address	IPv6 Gateway
pod-1/node-1	bdsol-aci37-apic1	Out-Of-Band	default	10.48.176.57/24	10.48.176.1	::	::
pod-1/node-101	S1P1-Leaf101	Out-Of-Band	default	10.48.176.70/24	10.48.176.1	::	::
pod-1/node-102	S1P1-Leaf102	Out-Of-Band	default	10.48.176.71/24	10.48.176.1	::	::
pod-1/node-2	bdsol-aci37-apic2	Out-Of-Band	default	10.48.176.58/24	10.48.176.1	::	::
pod-1/node-201	S1P1-Spine201	Out-Of-Band	default	10.48.176.74/24	10.48.176.1	::	::
pod-1/node-202	S1P1-Spine202	Out-Of-Band	default	10.48.176.75/24	10.48.176.1	::	::
pod-1/node-301	S1P2-Leaf301	Out-Of-Band	default	10.48.176.72/24	10.48.176.1	::	::
pod-1/node-302	S1P2-Leaf302	Out-Of-Band	default	10.48.176.73/24	10.48.176.1	::	::
pod-1/node-401	S1P2-Spine401	Out-Of-Band	default	10.48.176.76/24	10.48.176.1	::	::
pod-1/node-402	S1P2-Spine402	Out-Of-Band	default	10.48.176.77/24	10.48.176.1	::	::
pod-2/node-3	bdsol-aci37-apic3	Out-Of-Band	default	10.48.176.59/24	10.48.176.1	::	::

The out-of-band EPG should be present under the Node Management EPGs folder.

### Out-of-band EPG - default

The screenshot shows the Cisco APIC interface for configuring an Out-of-Band EPG. The 'mgmt' tenant is selected in the top navigation bar. In the left-hand navigation pane, the 'Node Management EPGs' folder is expanded, and 'Out-of-Band EPG - default' is highlighted with a red box. The main configuration area shows the following details:

- Name:** default
- Tags:** (empty field with a note: "enter tags separated by comma")
- Configuration Issues:** (empty)
- Configuration State:** applied
- Class ID:** 16387
- QoS Class:** Unspecified

Below these details is a table titled "Provided Out-of-Band Contracts":

OOB Contract	Tenant	Type	QoS Class	State
OOB-default	mgmt	oobbrc-OOB-default	Unspecified	formed

At the bottom of the configuration page, there are three buttons: "Show Usage", "Reset", and "Submit".

The contracts which govern which management services are provided from the out-of-band EPG are special contracts that are configured in the out-of-band contracts folder.

### Out-of-band contract

The screenshot shows the Cisco APIC interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Tenants' tab is active, and the 'mgmt' tenant is selected. The left sidebar shows a tree view of the 'mgmt' tenant, with 'Out-Of-Band Contracts' expanded and 'OOB-default' selected. The main content area displays the configuration for the 'OOB-default' contract. The configuration includes a 'Name' field with the value 'OOB-default', a 'Description' field with the value 'optional', and a 'Reverse Filter Ports' checkbox that is checked. Below this is a 'Filters' table with the following data:

Name	Tenant	State	Action
default	common	formed	Permit

At the bottom of the configuration page, there are buttons for 'Show Usage', 'Reset', and 'Submit'.

Next, verify the External Management Network Instance Profile is created and that the correct out-of-band contract is configured as the 'Consumed Out-Of-Band Contract'.

### External Management Network Instance Profile

The screenshot shows the APIC interface for configuring an External Management Network Instance Profile. The 'mgmt' tenant is selected, and the 'External Management Network Instance Profile - default' page is displayed. The 'Consumed Out-of-Band Contracts' table shows a contract named 'OOB-default' for the 'mgmt' tenant, with a type of 'oobbrc-OOB-default' and a state of 'formed'.

Out-of-Band Contract	Tenant	Type	QoS Class	State
OOB-default	mgmt	oobbrc-OOB-default	Unspecified	formed

The next items to verify are the interface state and cabling, and then the connectivity to the gateway.

- To check if the oobmgmt interface is up, enter 'ifconfig oobmgmt' on the APIC CLI. Verify that the interface flags are 'UP' and 'RUNNING', that the correct IP address is configured, and that packets are increasing in the RX and TX counters. If any checks are missing, then verify the correct cables are being used and that they are connected to the correct physical management ports on the APIC. The management ports will be labelled Eth1-1 and Eth1-2 and recent hardware have oobmgmt stickers to indicate the out-of-band interface. For more information about the physical out-of-band mgmt ports on the back of an

APIC, please refer to the section "Initial fabric setup" in chapter "Fabric discovery".

```

apic1# ifconfig oobmgmt
oobmgmt: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.4.20 netmask 255.255.255.0 broadcast 192.168.4.255
inet6 fe80::7269:5aff:feca:2986 prefixlen 64 scopeid 0x20<link>
ether 70:69:5a:ca:29:86 txqueuelen 1000 (Ethernet)
RX packets 295605 bytes 766226440 (730.7 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 253310 bytes 38954978 (37.1 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

- To check the network connectivity through the OOB, use ping to test the path of the packet through the out-of-band network.

```

apic1# ping 192.168.4.1
PING 192.168.4.1 (192.168.4.1) 56(84) bytes of data.
64 bytes from 192.168.4.1: icmp_seq=1 ttl=255 time=0.409 ms
64 bytes from 192.168.4.1: icmp_seq=2 ttl=255 time=0.393 ms
64 bytes from 192.168.4.1: icmp_seq=3 ttl=255 time=0.354 ms

```

Using traceroute in the bash shell on the APIC, trace the connectivity to the end user. If the traceroute is incomplete, login to this device (if accessible) and ping the oobmgmt interface and ping the host. Depending on which direction fails, troubleshoot the issue as a traditional networking problem.

Traceroute works by sending UDP packets with an increasing TTL, starting with 1. If a router receives the packet with TTL 1 and needs to route it, it drops the frame and sends back an ICMP unreachable message to the sender. Each hop is sent 3 UDP packets at the current TTL, and asterisks represent attempts where an ICMP unreachable / TTL Exceeded packet was not received. These 3 asterisk blocks are expected in most networks as some routing devices have ICMP unreachable / TTL Exceeded messages disabled, so when they receive TTL 1 packets that they need to route, they simply drop the packet and do not send the message back to the sender.

```

apic1# bash
admin@apic1:~> traceroute 10.55.0.16
traceroute to 10.55.0.16 (10.55.0.16), 30 hops max, 60 byte packets
 1 192.168.4.1 (192.168.4.1) 0.368 ms 0.355 ms 0.396 ms
 2 * * *
 3 * * *
 4 10.0.255.221 (10.0.255.221) 6.419 ms 10.0.255.225 (10.0.255.225) 6.447 ms *
 5 * * *
 6 * * *
 7 10.55.0.16 (10.55.0.16) 8.652 ms 8.676 ms 8.694 ms

```

The leaf switches have access to the `tcpdump` command, which can be used to verify which packets are traversing the `oobmgmt` interface. The example below captures on `'eth0'`, which is the `oobmgmt` interface used on the leaf and spine switches, and uses `'-n'` option for `tcpdump` to give the IP addresses used instead of the DNS names, and then filtering specifically for NTP packets (UDP port 123). Recall that in the previous example the leaf is polling NTP server 172.18.108.14. Below, the user can verify that NTP packets are being transmitted via the out-of-band interface and also that the leaf is receiving a response from the server.

```

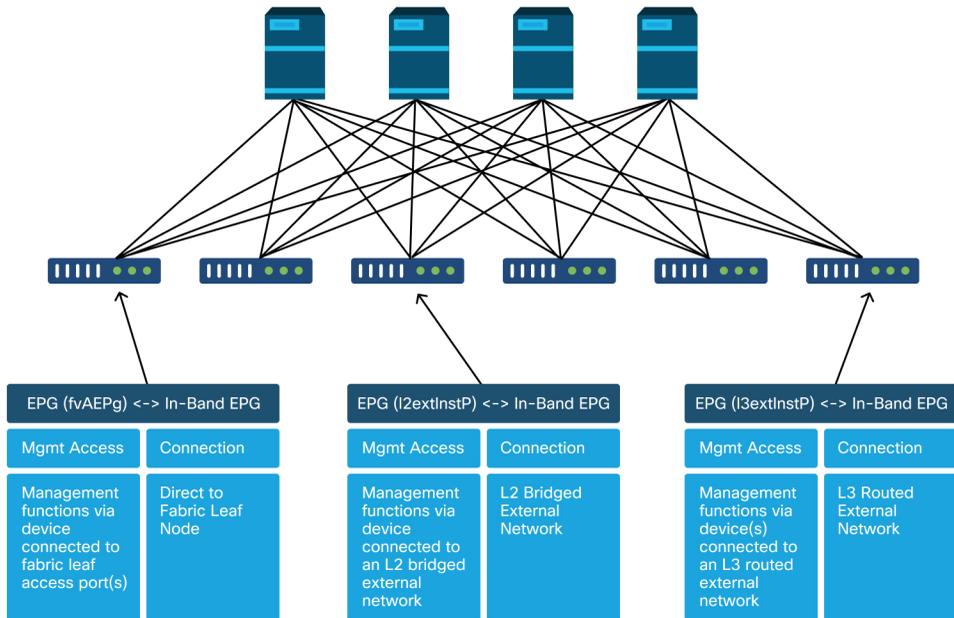
fab1-leaf101# tcpdump -n -i eth0 dst port 123
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
16:49:01.431624 IP 192.168.4.23.123 > 172.18.108.14.123: NTPv4, Client, length 48
16:49:01.440303 IP 172.18.108.14.123 > 192.168.4.23.123: NTPv4, Server, length 48

```

### In-band management configuration

The in-band management configuration requires specific considerations for Layer 2 or Layer 3 deployments. This example will only cover the Layer 3 deployment and troubleshooting.

In-band management configuration



Verify that there is a BD in the mgmt tenant with a subnet from which in-band node mgmt addresses will be allocated to the fabric nodes for in-band connectivity, and make sure that the L3Out is associated under the in-band management BD.

### Bridge Domain Subnet which will act as the in-band management gateway

The screenshot displays the Cisco APIC interface for configuring a Bridge Domain. The left sidebar shows the navigation tree with 'mgmt' selected, and 'Bridge Domains' > 'inb' highlighted. The main panel shows the 'Bridge Domain - inb' configuration page. The 'Policy' tab is selected, and the 'L3 Configurations' sub-tab is active. The 'Subnets' table is as follows:

Gateway Address	Scope	Primary IP Address	Virtual IP	Subnet Control
10.30.30.1/24	Advertised Externally	False	False	

The 'Associated L3 Outs' section shows the following configuration:

L3 Out
inbmgmt_l3out

Verify an in-band node management EPG is present. As per screenshot below, the in-band EPG names are denoted in the GUI with the prefix 'inb-'. Verify the in-band EPG encap VLAN is associated correctly with a VLAN pool.

The encapsulation VLAN configured in the in-band management EPG needs to be allowed by Access Policies: 'inb mgmt EPG encap VLAN > VLAN Pool > Domain > AEP > Interface Policy Group > Leaf Interface Profile > Switch Profile'. If the supporting access policies are not configured, a fault with code F0467 will be raised as per below screenshot.

## Fault F0467 - inb EPG

8589935303

ID: 8589935303

Description: Fault delegate: Configuration failed for uni/tn-mgmt/mgmtmp-default/inb-inbmgmt due to Invalid VLAN Configuration, debug message: i  
vlan-300STP Segment Id not present for Encap. Either the EpG is not associated with a domain or the domain does not have this vlan :

Severity: minor

Affected Object: [uni/tn-mgmt/mgmtmp-default/inb-inbmgmt](#) 

Generated From: topology/pod-1/node-101/local/svc-policyelem-id-0/uni/epp/inb-[uni/tn-mgmt/mgmtmp-default/inb-inbmgmt]/nwissues

Created: 2019-10-03T02:23:04.637+00:00

Code: F0467

Type: Config

Cause: configuration-failed

Change Set:

Action: deletion

Domain: Tenant

Life Cycle:

Count Occurred: 1

Event Status: false

Verify that the bridge domain is the same as the one created above for the in-band subnet. Lastly, verify that there is a Provided Contract configured on the in-band management EPG, which is consumed by the external EPG.

## In-band EPG

The screenshot displays the Cisco APIC web interface. At the top, the navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Tenants' tab is active, and the breadcrumb path is 'ALL TENANTS | Add Tenant | Tenant Search: name of descr | common | infra | mgmt | Ecommerce'. The left sidebar shows a tree view under 'mgmt' with 'Node Management EPGs' expanded, highlighting 'In-Band EPG - inb\_mgmt'.

The main content area is titled 'In-Band EPG - inb\_mgmt' and features tabs for 'Policy', 'Stats', 'Health', 'Faults', and 'History'. The 'Policy' tab is selected, showing a configuration page with a green status bar at 100%. The configuration details are as follows:

- Properties:** Name: inb\_mgmt; Tags: (empty); Encap: vlan-300 (with a note 'e.g., vlan-1').
- Configuration Issues:** Configuration State: applied; Class ID: 32770.
- QoS Class:** Unspecified
- Bridge Domain:** inb
- Resolved Bridge Domain:** inb
- Provided Contracts:** A table listing the contract 'default' from tenant 'common' with type 'Contract', QoS Class 'Unspecified', Match Type 'AtleastOne', and State 'formed'.

At the bottom right, there are three buttons: 'Show Usage', 'Reset', and 'Submit'.

### External EPG Instance Profile

The screenshot shows the Cisco APIC interface for the 'mgmt' tenant. The left sidebar shows the navigation tree with 'Inband-Out' under 'External EPGs' highlighted. The main content area displays the 'External EPG Instance Profile - Inband-Out' configuration page. The 'Contracts' tab is selected, showing a table of contract details.

Name	Tenant	Type	QoS Class	State
default	common	Contract	Unspecified	formed

Similar to out-of-band, fabric node in-band mgmt IP addresses can be statically assigned or dynamically assigned from a pre-selected range. Verify the addresses applied for type in-band match the previous BD subnet that was configured. Also verify that the default gateway is correct.

## Static Node Management Addresses

The screenshot shows the Cisco APIC interface. The 'mgmt' tab is selected in the top navigation bar. The left sidebar shows the 'mgmt' menu with 'Static Node Management Addresses' highlighted. The main content area displays a table titled 'Static Node Management Addresses'.

Node ID	Name	Type	EPG	IPv4 Address	IPv4 Gateway	IPv6 Address	IPv6 Gateway
pod-1/node-1	bdsol-aci37-apic1	Out-Of-Band	default	10.48.176.57/24	10.48.176.1	::	::
pod-1/node-101	S1P1-Leaf101	In-Band	inb_mg...	10.30.30.101/24	10.30.30.1	::	::
pod-1/node-101	S1P1-Leaf101	Out-Of-Band	default	10.48.176.70/24	10.48.176.1	::	::
pod-1/node-102	S1P1-Leaf102	Out-Of-Band	default	10.48.176.71/24	10.48.176.1	::	::
pod-1/node-2	bdsol-aci37-apic2	Out-Of-Band	default	10.48.176.58/24	10.48.176.1	::	::
pod-1/node-201	S1P1-Spine201	Out-Of-Band	default	10.48.176.74/24	10.48.176.1	::	::
pod-1/node-202	S1P1-Spine202	Out-Of-Band	default	10.48.176.75/24	10.48.176.1	::	::
pod-1/node-301	S1P2-Leaf301	Out-Of-Band	default	10.48.176.72/24	10.48.176.1	::	::
pod-1/node-302	S1P2-Leaf302	Out-Of-Band	default	10.48.176.73/24	10.48.176.1	::	::
pod-1/node-401	S1P2-Spine401	Out-Of-Band	default	10.48.176.76/24	10.48.176.1	::	::
pod-1/node-402	S1P2-Spine402	Out-Of-Band	default	10.48.176.77/24	10.48.176.1	::	::
pod-2/node-3	bdsol-aci37-apic3	Out-Of-Band	default	10.48.176.59/24	10.48.176.1	::	::

If everything has been configured correctly, and there are no faults in any above-mentioned section, the next step is to ping between the switches and/or APICs to verify that in-band connectivity is working correctly inside ACI.

The spine nodes will not respond to ping on the in-band as they use loopback interfaces for connectivity which do not respond to ARP.

The in-band interface used on the leaf switches is `kpm_inb`. Using a similar `tcpdump` capture, verify the packet is egressing the in-band CPU interface.

```
fab2-leaf101# tcpdump -n -i kpm_inb dst port 123
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes
16:46:50.431647 IP 10.30.30.3.123 > 172.18.108.14.123: NTPv4, Client, length 48
16:47:19.431650 IP 10.30.30.3.123 > 172.18.108.15.123: NTPv4, Client, length 48
```

Verify that the SVI used for in-band is `'protocol-up/link-up/admin-up'`.

```
fab1-leaf101# show ip interface vrf mgmt:inb-vrf
IP Interface Status for VRF "mgmt:inb-vrf"
vlan16, Interface status: protocol-up/link-up/admin-up, iof: 4, mode: pervasive
  IP address: 10.30.30.1, IP subnet: 10.30.30.0/24 secondary
  IP address: 10.30.30.3, IP subnet: 10.30.30.0/24
  IP broadcast address: 255.255.255.255
  IP primary address route-preference: 0, tag: 0
```

# Pod Policies – BGP RR / Date&Time / SNMP

## Introduction

Management services such as BGP RR, Date & Time and SNMP are applied on the system using a Pod Policy Group. A Pod Policy Group governs a group of Pod Policies related to essential functions of an ACI Fabric. These Pod Policies relate to the following components, many of which are provisioned in an ACI fabric by default.

### Pod Policies

Pod Policy	Requires Manual Config
<b>Date &amp; Time</b>	Yes
<b>BGP Route Reflector</b>	Yes
<b>SNMP (server network management protocol)</b>	Yes
<b>ISIS</b>	No
<b>COOP</b>	No
<b>Management Access</b>	No
<b>MAC Sec</b>	Yes

Even in a single ACI fabric, the Pod Policy Group and Pod Profile need to be configured. This is not specific to a Multi-Pod or even a Multi-Site deployment. The requirement applies to **all** ACI deployment types.

This chapter focuses on these essential Pod Policies and how to verify they're applied correctly.

## Date & Time policy

### About the Date & Time policy

Time synchronization plays a critical role in the ACI fabric. From validating certificates, to keeping log timestamps in APICs and switches consistent, it is best practice to sync the nodes in the ACI fabric to one or more reliable time sources using NTP.

In order to properly have the nodes synchronized to an NTP server provider, there's a dependency to assign nodes with management addresses. This can be done under the management tenant using either Static Node Management Addresses or Management Node Connectivity Groups.

### Workflow

#### 1. Verify if Node Management Addresses are assigned to all nodes

#### Management tenant - Node Management Addresses

The screenshot shows the APIC interface for the 'mgmt' tenant. The left sidebar contains a navigation menu with 'Static Node Management Addresses' highlighted. The main content area displays a table of 'Static Node Management Addresses' with the following data:

Node ID	Name	Type	EPG	IPv4 Address	IPv4 Gateway	IPv6 Address	IPv6 Gateway
pod-1/node-101	S1P1-Leaf101	Out-Of-Band	default	10.48.176.70/24	10.48.176.1	::	::
pod-1/node-102	S1P1-Leaf102	Out-Of-Band	default	10.48.176.71/24	10.48.176.1	::	::
pod-1/node-201	S1P1-Spine201	Out-Of-Band	default	10.48.176.74/24	10.48.176.1	::	::
pod-1/node-202	S1P1-Spine202	Out-Of-Band	default	10.48.176.75/24	10.48.176.1	::	::
pod-1/node-301	S1P2-Leaf301	Out-Of-Band	default	10.48.176.72/24	10.48.176.1	::	::
pod-1/node-302	S1P2-Leaf302	Out-Of-Band	default	10.48.176.73/24	10.48.176.1	::	::
pod-1/node-401	S1P2-Spine401	Out-Of-Band	default	10.48.176.76/24	10.48.176.1	::	::
pod-1/node-402	S1P2-Spine402	Out-Of-Band	default	10.48.176.77/24	10.48.176.1	::	::

## 2. Verify if an NTP server has been configured as an NTP provider

If there are multiple NTP providers, flag at least one of them as the preferred time source using the 'Preferred' checkbox as per the figure below.

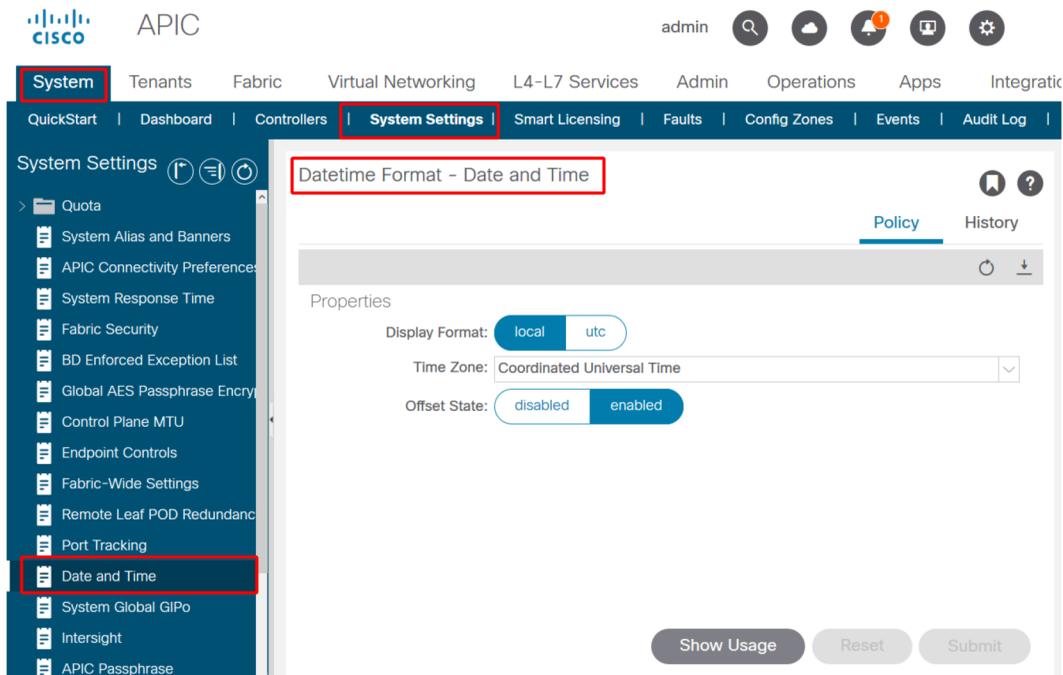
### NTP Provider/Server under Date and Time Pod Policy

The screenshot displays the Cisco APIC web interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Fabric' tab is selected, and the 'Fabric Policies' sub-tab is active. The left-hand navigation pane shows a tree structure under 'Policies' with 'Pod' expanded to 'Date and Time', where 'NTP Server 10.48.37.151' is highlighted. The main content area shows the configuration for 'Providers - NTP Server 10.48.37.151'. The 'Host Name/IP Address' field is set to '10.48.37.151'. The 'Description' field contains 'optional'. The 'Preferred' checkbox is checked. The 'Minimum Polling Interval' is set to 4 and the 'Maximum Polling Interval' is set to 6. At the bottom, there are 'Show Usage', 'Reset', and 'Submit' buttons.

### 3. Verify the Date and Time format under System Settings

The figure below shows an example whereby the Date and Time format has been set to UTC.

Date and Time setting under System Settings



#### 4. Verify the operational Sync Status of the NTP provider for all nodes

As shown in the figure below, the Sync Status column should show 'Synced to Remote NTP Server'. Be aware that it can take several minutes for the Sync Status to converge properly to the .Synced to Remote NTP Server. status.

#### NTP Provider/Server Sync Status

The screenshot shows the APIC interface with the following elements:

- Top navigation: System, Tenants, **Fabric**, Virtual Networking, L4-L7 Services, Admin, Operations, Apps, Integrator.
- Left sidebar: Policies > Pod > **NTP Server 10.48.37.151**.
- Main content area: Providers - NTP Server 10.48.37.151.
 

Name	Switch	VRF	Preferred	Sync Status
10.48.37.151	Node-101	management	True	Synced to Remote NTP Server
10.48.37.151	Node-103	management	True	Synced to Remote NTP Server
10.48.37.151	Node-104	management	True	Synced to Remote NTP Server
10.48.37.151	Node-105	management	True	Synced to Remote NTP Server
10.48.37.151	Node-102	management	True	Synced to Remote NTP Server
10.48.37.151	Node-201	management	True	Synced to Remote NTP Server
10.48.37.151	Node-106	management	True	Synced to Remote NTP Server
10.48.37.151	Node-202	management	True	Synced to Remote NTP Server

Alternatively, CLI methods can be used on the APICs and the switches to verify correct time sync against the NTP Server.

## APIC - NX-OS CLI

The 'refId' column below shows the NTP Servers next time source depending on the stratum.

```

apic1# show ntpq
nodeid  remote
  auth  delay  offset  jitter  refid          st  t  when  poll  reach
-----
-----
1      *  10.48.37.151          173.38.201.115  2  u  25   64   377
  none  0.214  -0.118  0.025
2      *  10.48.37.151          173.38.201.115  2  u  62   64   377
  none  0.207  -0.085  0.043
3      *  10.48.37.151          173.38.201.115  2  u  43   64   377
  none  0.109  -0.072  0.030

apic1# show clock
Time : 17:38:05.814 UTC Wed Oct 02 2019

```

## APIC - Bash

```

apic1# bash
admin@apic1:~> date
Wed Oct  2 17:38:45 UTC 2019

```

## Switch

Use the 'show ntp peers' command to make sure the NTP provider configuration has been properly pushed to the switch.

```

leaf1# show ntp peers
-----
Peer IP Address          Serv/Peer Prefer KeyId  Vrf
-----
10.48.37.151             Server  yes  None  management

leaf1# show ntp peer-status
Total peers : 1
* - selected for sync, + - peer mode(active),

```

```

-- peer mode(passive), = - polled in client mode
remote                local                st poll reach delay vrf
-----
*10.48.37.151         0.0.0.0                2 64 377 0.000 management

```

The '\*' character is essential here as it governs whether the NTP server is actually being used for sync.

Verify the number of packets sent/received in the following command to make sure ACI nodes have reachability to the NTP server.

```

leaf1# show ntp statistics peer ipaddr 10.48.37.151
...
packets sent:         256
packets received:    256
...

```

## BGP Route Reflector policy

### About the BGP Route Reflector policy

An ACI fabric uses multi-protocol BGP (MP-BGP) and, more specifically, iBGP VPNv4 between leaf and spine nodes to exchange tenant routes received from external routers (connected on L3Outs). To avoid a full mesh iBGP peer topology, the spine nodes reflect VPNv4 prefixes received from a leaf to other leaf nodes in the fabric.

Without the BGP Route Reflector (BGP RR) Policy, no BGP instance will be created on the switches and BGP VPNv4 sessions won't be established. In a Multi-Pod deployment, each Pod requires at least one spine configured as a BGP RR and essentially more than one for redundancy.

As a result, the BGP RR Policy is an essential piece of configuration in every ACI Fabric. The BGP RR Policy also contains the ASN the ACI Fabric uses for the BGP process on each switch.

## Workflow

### 1. Verify if the BGP RR Policy has an ASN and at least one spine configured

The example below refers to a single Pod deployment.

#### BGP Route Reflector Policy under System Settings

The screenshot shows the APIC interface for configuring a BGP Route Reflector Policy. The 'System' and 'System Settings' tabs are highlighted. The configuration page for 'BGP Route Reflector Policy - BGP Route Reflector' is displayed, showing the following details:

- Name:** default
- Description:** optional
- Autonomous System Number:** 65001
- Route Reflector Nodes:**

Pod ID	Node ID	Node Name	Description
1	201	bdsol-aci12-spine1	
1	202	bdsol-aci12-spine2	

Buttons at the bottom include 'Show Usage', 'Reset', and 'Submit'.

## 2. Verify if the BGP RR Policy is applied under the Pod Policy Group

Apply a default BGP RR Policy under the Pod Policy Group. Even if the entry is blank, the default BGP RR Policy will be applied as part of the Pod Policy Group.

### BGP Route Reflector Policy applied under Pod Policy Group

The screenshot displays the APIC (Application Policy Infrastructure Controller) interface. The top navigation bar includes the Cisco logo, the text "APIC", and user information "admin" with search, notifications, and settings icons. The main navigation menu contains "System", "Tenants", "Fabric" (highlighted in red), "Virtual Networking", "L4-L7 Services", "Admin", "Operations", and "Apps". Below this, a sub-menu shows "Inventory", "Fabric Policies" (highlighted in red), and "Access Policies".

The left sidebar lists various policy categories: "Policies" (with navigation icons), "Quick Start", "Pods" (highlighted in red), "Policy Groups", "All", "Profiles", "Switches", "Modules", "Interfaces", "Policies", and "Tags".

The main content area is titled "Pod Policy Group - All" and has tabs for "Policy", "Faults", and "History". The "Policy" tab is active. Below the tabs, there are icons for refresh, download, and edit. The "Properties" section contains the following configuration items:

- Name: All
- Description: optional
- Date Time Policy: default
- Resolved Date Time Policy: default
- ISIS Policy: select a value
- Resolved ISIS Policy: default
- COOP Group Policy: select a value
- Resolved COOP Group Policy: default
- BGP Route Reflector Policy: default (highlighted in red)

At the bottom of the configuration area, there are three buttons: "Show Usage", "Reset", and "Submit".

### 3. Verify if the Pod Policy Group is applied under the Pod Profile

Pod Policy Group applied under the Pod Profile

The screenshot shows the APIC interface with the 'Fabric Policies' menu selected. The 'Pod Profile - default' configuration page is displayed, showing the 'Policy' tab. The 'Pod Selectors' table is as follows:

Name	Type	Blocks	Policy Group
default	ALL	ALL	All

### 4. Log into a spine and verify if the BGP Process is running with established VPN4 peer sessions

```
spine1# show bgp process vrf overlay-1

BGP Process Information
BGP Process ID       : 26660
BGP Protocol Started, reason : configuration
BGP Protocol Tag           : 65001
BGP Protocol State        : Running
BGP Memory State     : OK
BGP asformat         : asplain
Fabric S00            : S00:65001:33554415
Multisite S00        : S00:65001:16777199
```

```

Pod S00                               : S00:1:1
...

Information for address family VPNv4 Unicast in VRF overlay-1
Table Id                               : 4
Table state                           : UP
Table refcount                          : 9
Peers      Active-peers  Routes    Paths    Networks  Aggregates
7          6              0        0        0         0

Redistribution
  None

Wait for IGP convergence is not configured
Additional Paths Selection route-map interleak_rtmmap_golf_rtmmap_path_advertise_all
Is a Route-reflector

NextHop trigger-delay
  critical 500 ms
  non-critical 5000 ms

Information for address family VPNv6 Unicast in VRF overlay-1
Table Id                               : 80000004
Table state                             : UP
Table refcount                          : 9
Peers      Active-peers  Routes    Paths    Networks  Aggregates
7          6              0        0        0         0

Redistribution
  None

Wait for IGP convergence is not configured
Additional Paths Selection route-map interleak_rtmmap_golf_rtmmap_path_advertise_all
Is a Route-reflector

NextHop trigger-delay
  critical 500 ms
  non-critical 5000 ms
...

Wait for IGP convergence is not configured
Is a Route-reflector

NextHop trigger-delay
  critical 500 ms
  non-critical 5000 ms

```

As shown above, MP-BGP between leaf and spine nodes carries only VPNv4 and VPNv6 address families. The IPv4 address family is used in MP-BGP only on leaf nodes.

The BGP VPNv4 and VPNv6 sessions between spine and leaf nodes can also be easily observed using the following command.

```
spine1# show bgp vpnv4 unicast summary vrf overlay-1
BGP summary information for VRF overlay-1, address family VPNv4 Unicast
BGP router identifier 10.0.136.65, local AS number 65001
BGP table version is 15, VPNv4 Unicast config peers 7, capable peers 6
0 network entries and 0 paths using 0 bytes of memory
BGP attribute entries [0/0], BGP AS path entries [0/0]
BGP community entries [0/0], BGP clusterlist entries [0/0]
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
10.0.136.64	4	65001	162	156	15	0	0	02:26:00	0
10.0.136.67	4	65001	154	154	15	0	0	02:26:01	0
10.0.136.68	4	65001	152	154	15	0	0	02:26:00	0
10.0.136.69	4	65001	154	154	15	0	0	02:26:01	0
10.0.136.70	4	65001	154	154	15	0	0	02:26:00	0
10.0.136.71	4	65001	154	154	15	0	0	02:26:01	0

```
spine1# show bgp vpnv6 unicast summary vrf overlay-1
BGP summary information for VRF overlay-1, address family VPNv6 Unicast
BGP router identifier 10.0.136.65, local AS number 65001
BGP table version is 15, VPNv6 Unicast config peers 7, capable peers 6
0 network entries and 0 paths using 0 bytes of memory
BGP attribute entries [0/0], BGP AS path entries [0/0]
BGP community entries [0/0], BGP clusterlist entries [0/0]
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
10.0.136.64	4	65001	162	156	15	0	0	02:26:11	0
10.0.136.67	4	65001	155	155	15	0	0	02:26:12	0
10.0.136.68	4	65001	153	155	15	0	0	02:26:11	0
10.0.136.69	4	65001	155	155	15	0	0	02:26:12	0
10.0.136.70	4	65001	155	155	15	0	0	02:26:11	0
10.0.136.71	4	65001	155	155	15	0	0	02:26:12	0

Note the 'Up/Down' column from the above output. It should list a duration time which denotes the time the BGP session has been established. Also note in the example the 'PfxRcd' column shows 0 for each BGP VPNv4/VPNv6 peer as this ACI Fabric has no L3Outs configured yet and as such no external routes/prefixes are exchanged between leaf and spine nodes.

## 5. Log into a leaf and verify if the BGP Process is running with established VPN4 peer sessions

```

leaf1# show bgp process vrf overlay-1

BGP Process Information
BGP Process ID           : 43242
BGP Protocol Started, reason: : configuration
BGP Protocol Tag         : 65001
BGP Protocol State       : Running
...

leaf1# show bgp vpnv4 unicast summary vrf overlay-1
BGP summary information for VRF overlay-1, address family VPNv4 Unicast
BGP router identifier 10.0.136.64, local AS number 65001
BGP table version is 7, VPNv4 Unicast config peers 2, capable peers 2
0 network entries and 0 paths using 0 bytes of memory
BGP attribute entries [0/0], BGP AS path entries [0/0]
BGP community entries [0/0], BGP clusterlist entries [0/0]

Neighbor      V      AS MsgRcvd MsgSent  TblVer  InQ  OutQ  Up/Down  State/PfxRcd
10.0.136.65   4  65001   165    171     7    0    0 02:35:52 0
10.0.136.66   4  65001   167    171     7    0    0 02:35:53 0

```

The above command outputs show an amount of BGP VPNv4 sessions equal to the number of spine nodes present in the ACI Fabric. This differs from the spine nodes because they establish sessions to each leaf and the other route reflector spine nodes.

## SNMP

### About the SNMP Policy (Pod Policies) and differentiating ingress/egress SNMP traffic flows

It is important to clarify from the start which specific subset of SNMP functions this section covers. SNMP functions in an ACI fabric either relate to the SNMP Walk function or the SNMP Trap function. The important distinction here is that SNMP Walk governs **ingress** SNMP traffic flows on UDP port 161 whereas SNMP Trap governs **outgoing** SNMP traffic flows with an SNMP Trap server listening on UDP port 162.

Ingress management traffic on ACI nodes require the Node Management EPGs (either in-band or out-of-band) to provide the necessary contracts to allow the traffic to flow. As such this also applies to ingress SNMP traffic flows.

This section will cover the ingress SNMP traffic flows (SNMP Walks) into ACI nodes (APICs and switches). It will not cover the egress SNMP traffic flows (SNMP Traps) as

that would expand the scope of this section into Monitoring Policies and Monitoring Policy dependencies (i.e. Monitoring Policy scope, Monitoring Packages, etc.).

This section also won't cover which SNMP MIBs are supported by ACI. That information is available on the Cisco CCO website in the following link: <https://www.cisco.com/c/da/m/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/mib/list/mib-support.html>

## Workflow

### 1. SNMP Pod Policy – Verify if a Client Group Policy is configured

Make sure at least a single SNMP Client is configured as part of the Client Group Policy as per screenshots below.

#### Pod Policies – SNMP Policy – Client Group Policies

The screenshot displays the Cisco ACI GUI configuration page for an SNMP Policy. The navigation menu on the left shows the path: Fabric Policies > Policies > Pod > SNMP > default. The main configuration area is titled 'SNMP Policy - default' and includes the following fields:

- Name: default
- Description: optional
- Admin State: Enabled
- Contact: (empty field)
- Location: (empty field)

Below the configuration fields is a table for 'Client Group Policies':

Name	Description	Client Entries	Associated Management EPG
snmpClientGrpProf		10.155.0.153	default (Out-of-Band)

At the bottom of the page, there are buttons for 'Show Usage', 'Reset', and 'Submit'.

Pod Policies – SNMP Policy – Client Group Policies

SNMP Client Group Profile - snmpClientGrpProf

Policy History

Refresh Download Edit

Properties

Name: snmpClientGrpProf

Description: optional

Associated Management EPG: default (Out-of-Band)

Client Entries:

Name	Address
Server01	10.155.0.153

## 2. SNMP Pod Policy – Verify if at least one Community Policy is configured

### Pod Policies – SNMP Policy – Community Policies

The screenshot displays the network management interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integration'. The left sidebar shows a navigation tree with 'Policies' expanded, and 'SNMP' > 'default' selected. The main panel shows the 'SNMP Policy - default' configuration page. The 'Community Policies' table is highlighted with a red box, showing one entry: 'my-secret-SNMP-community'. The table has columns for 'Name' and 'Description'. Below the table, there is a 'Trap Forward Servers' section with columns for 'IP Address' and 'Port'. At the bottom, there are buttons for 'Show Usage', 'Reset', and 'Submit'.

Name	Description
my-secret-SNMP-community	

### 3. SNMP Pod Policy – Verify if the Admin State is set to 'Enabled'

#### Pod Policies – SNMP Policy – Admin State

The screenshot shows the configuration page for an SNMP Policy. The breadcrumb path is System > Tenants > Fabric > Fabric Policies > Access Policies > SNMP Policy - default. The 'Admin State' is set to 'Enabled'. Below the configuration fields is a table of Client Group Policies.

Name	Description	Client Entries	Associated Management EPG
snmpClientGrpProf		10.155.0.153	default (Out-of-Ban...

**4. Management tenant – verify if the OOB EPG is providing an OOB Contract allowing UDP port 161**

The OOB EPG governs connectivity into the APIC and switch OOB management ports. As such it affects all traffic flows ingressing into the OOB ports.

Make sure the contract which is provided here includes all necessary management services instead of just SNMP. For example: it also needs to include at least SSH (TCP port 22). Without this it is not possible to log into the switches using SSH. Please note this does not apply to APICs as they have a mechanism to allow SSH, HTTP, HTTPS to prevent users from being locked up completely.

Management tenant – OOB EPG – provided OOB Contract

The screenshot displays the Cisco APIC interface for configuring an Out-of-Band EPG. The top navigation bar shows the 'mgmt' tenant is active. The left sidebar lists various configuration categories, with 'Node Management EPGs' and 'Out-of-Band EPG - default' highlighted. The main configuration area shows the 'Out-of-Band EPG - default' configuration with the 'Policy' tab selected. A table of 'Provided Out-of-Band Contracts' is visible, containing the following data:

OOB Contract	Tenant	Type	QoS Class	State
snmp-walk-oob-contract	mgmt	oobrc-snmp-walk-oob-contract	Unspecified	formed

## 5. Management tenant – verify if the OOB Contract is present and has a filter allowing UDP port 161

### Management tenant – OOB EPG – Provided OOB Contract

The screenshot displays the Cisco APIC interface for the 'mgmt' tenant. The left-hand navigation pane shows the hierarchy: mgmt > Contracts > Out-Of-Band Contracts > snmp-walk-oob-subject. The main content area shows the configuration for the 'Contract Subject - snmp-walk-oob-subject'. The 'General' tab is active, showing the following details:

- Name: snmp-walk-oob-subject
- Description: optional
- Reverse Filter Ports:
- Filters: A table listing the configured filters.

Name	Tenant	State	Action
snmp-walk-filter	mgmt	formed	Permit

At the bottom of the configuration page, there are buttons for 'Show Usage', 'Reset', and 'Submit'.

In the figure below, it is not mandatory to just allow UDP port 161. A contract that has a filter allowing UDP port 161 in any manner is correct. This can even be a contract subject with the default filter from the common tenant. In our example, for clarity purposes, a specific filter was configured just for UDP port 161.

## Management tenant – OOB Contract – filter allowing UDP port 161

The screenshot shows the Cisco APIC interface for the 'mgmt' tenant. The 'Tenants' tab is selected, and the 'mgmt' tenant is active. The 'Filters' section is expanded, showing the 'snmp-walk-filter' filter. The filter is configured with the following properties:

- Name: snmp-walk-filter
- Alias: (empty)
- Description: optional
- Tags: (empty)
- Global Alias: (empty)

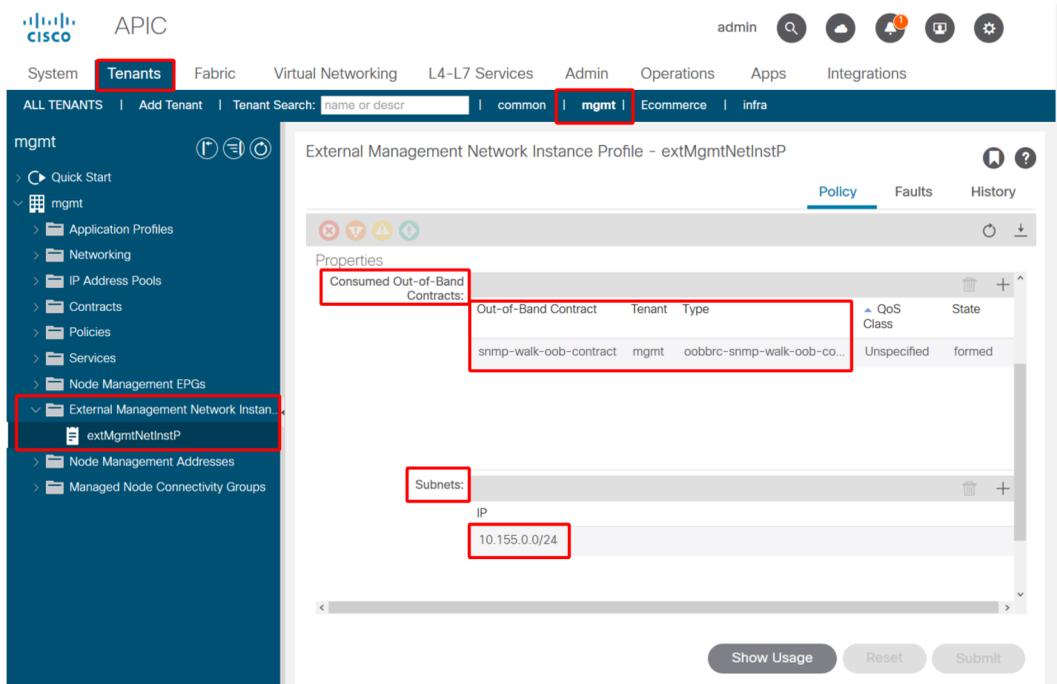
The filter entries table is as follows:

Name	Alias	EtherType	ARP Flag	IP Protocol	Match Only Fragme	Stateful	Source Port / Range	Destination Port / Range
sn...		IP		udp	False	False	unspecified	unspecified 161 161

## 6. Management tenant – verify if an External Management Network Instance Profile is present with a valid Subnet consuming the OOB Contract

The external management network instance profile (ExtMgmtNetInstP) represents external sources defined by the 'Subnets' in there that need to consume services reachable via the OOB EPG. So, the ExtMgmtNetInstP consumes the same OOB contract which is provided by the OOB EPG. This is the contract allowing UDP port 161. In addition, the ExtMgmtNetInstP also specifies the allowed subnet ranges that may consume the services provided by the OOB EPG.

Management tenant – ExtMgmtNetInstP with consumed OOB Contract and Subnet



As shown in the figure above, a CIDR-based subnet notation is required. The figure shows a specific /24 subnet. The requirement is that the subnet entries cover the SNMP Client Entries as configured in the SNMP Pod Policy (refer to Figure Pod Policies – SNMP Policy – Client Group Policies).

As mentioned earlier, please be careful to include all required external subnets to prevent other necessary management services from being locked out.

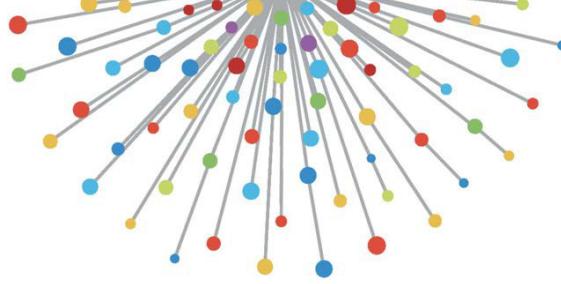
## 7. Log into a switch and perform a tcpdump to observe if SNMP Walk packets — UDP port 161 — are observed

If SNMP Walk packets are entering a switch through the OOB port, this means all necessary SNMP and OOB based policies/parameters have been properly configured. Hence, it's a proper verification method.

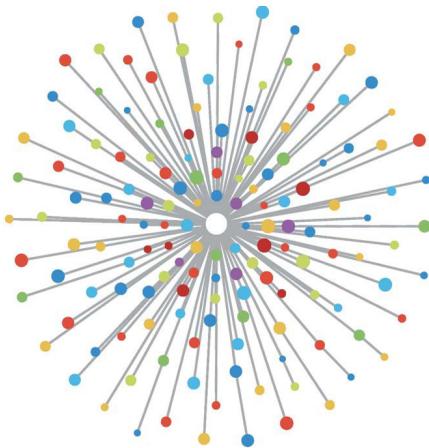
Tcpdump on the leaf nodes leverages their Linux shell and Linux netdevices. Hence, it's necessary to capture the packets on interface 'eth0' as per below example. In the example, an SNMP client is performing an SNMP Get request against OID .1.0.8802.1.1.2.1.1.1.0.

```
leaf1# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether f4:cf:e2:28:fc:ac brd ff:ff:ff:ff:ff:ff
    inet 10.48.22.77/24 brd 10.48.22.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::f6cf:e2ff:fe28:fcac/64 scope link
        valid_lft forever preferred_lft forever

leaf1# tcpdump -i eth0 udp port 161
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
22:18:10.204011 IP 10.155.0.153.63392 > 10.48.22.77.snmp: C=my-snmp-community GetNextRequest(28)
.iso.0.8802.1.1.2.1.1.1.0
22:18:10.204558 IP 10.48.22.77.snmp > 10.155.0.153.63392: C=my-snmp-community GetResponse(29)
.iso.0.8802.1.1.2.1.1.2.0=4
```



# Access policies



## Overview

How does the ACI administrator configure a VLAN on a port in the fabric? How does the ACI admin begin to address faults related to access policies? This section will explain how to troubleshoot issues related to fabric access policies.

Before jumping into troubleshooting scenarios, it is imperative that the reader have a good understanding of how access policies function and their relationships within the ACI Object Model. For this purpose, the reader can refer to both the "ACI Policy Model" and "APIC Management Information Model Reference" documents available on Cisco.com (<https://developer.cisco.com/site/apic-mim-ref-api/>).

The function of access policies is to enable specific configuration on a leaf switch's downlink ports. Before tenant policy is defined to allow traffic through an ACI fabric port, the related access policies should be in place.

Typically, access policies are defined when new leaf switches are added to the fabric, or a device is connected to ACI leaf downlinks; but depending on how dynamic an environment is, access policies could be modified during normal operation of the fabric. For example, to allow a new set of VLANs or add a new Routed Domain to fabric access ports.

The ACI access policies, though initially a bit intimidating, are extremely flexible and are designed to simplify the provisioning of configuration to a large scale SDN network in continuous evolution.

## Access policy configuration: Methodology

Access policies can be configured independently, i.e. by creating all the objects required independently, or can be defined through the numerous wizards provided by the ACI GUI.

Wizards are very helpful because they guide the user through the workflow and make sure all the required policies are in place.

### Access policies – Quick Start wizard

The screenshot displays the Cisco APIC interface for configuring Access Policies. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Fabric' section is active, showing 'Inventory', 'Fabric Policies', and 'Access Policies'. The 'Policies' sidebar on the left lists 'Quick Start', 'Switches', 'Modules', 'Interfaces', 'Policies', 'Pools', and 'Physical and External Domains'. The main content area is titled 'Quick Start' and contains a 'Summary' section explaining that access policies govern external access to the fabric. The 'Steps' section lists the following tasks:

- Configure in-band management access
- Configure out-of-band management access
- Create a CDP (or other) interface policy
- Create a traffic storm control policy
- Configure an interface, PC, and VPC** (highlighted in red)
- Quick configure port interface
- Configure port security

The 'See Also' section lists related configuration tasks:

- Physical Interface (Link Level)
- CDP
- LLDP
- LACP
- LACP Member
- Spanning Tree Interface
- Storm Control
- Port Security
- SPAN
- On-demand Diagnostics
- Attachable Entity Profile
- QoS
- DHCP Relay

The above image shows the Quick Start page where multiple wizards can be found.

Once an access policy is defined, the generic recommendation is to validate the policy by making sure all the associated objects do not show any fault.

For example, in the figure below, a Switch Profile has assigned an Interface Selector Policy that does not exist. An attentive user will easily be able to spot the **'missing-target'** state of the object and verify that a fault was flagged from the GUI:

### Leaf Profile – SwitchProfile\_101

The screenshot displays the Cisco APIC interface for configuring a Leaf Profile. The left sidebar shows the navigation tree under 'Fabric' > 'Access Policies' > 'Profiles', with 'SwitchProfile\_101' selected. The main content area shows the configuration for 'Leaf Profile - SwitchProfile\_101'. The 'Policy' tab is active, showing a table of associated interface selector profiles. The table has columns for Name, Description, and State. One entry is visible: 'Policy' with description 'SwitchProfile\_101' and state 'missing-target'. The 'missing-target' state is highlighted with a red rectangular box. At the bottom of the configuration area, there are buttons for 'Show Usage', 'Reset', and 'Submit'.

Name	Description	State
Policy	SwitchProfile_101	missing-target

## Leaf Profile – SwitchProfile\_101 – Fault

The screenshot displays the Cisco APIC interface. A 'Fault Properties' dialog box is open, showing details for a fault with code F1014. The dialog has tabs for 'General', 'Troubleshooting', and 'History', with 'General' selected. The fault details are as follows:

- Fault Code: F1014
- Severity: warning
- Last Transition: 2019-10-28T11:23:11.665+00:00
- Lifecycle: Raised
- Affected Object: uni/infra/nprof-SwitchProfile\_101/rsaccPortP-[uni/infra/accportprof-Policy]
- Description: Failed to form relation to MO uni/infra/accportprof-Policy of class infraAccPortP
- Type: Config
- Cause: resolution-failed
- Change Set: state (Old: formed, New: missing-target)
- Created: 2019-10-28T11:23:11.665+00:00
- Code: F1014
- Number of Occurrences: 1
- Original Severity: warning
- Previous Severity: warning
- Highest Severity: warning

The background interface shows a navigation tree on the left with 'Policies' selected. On the right, there is a table of faults with a 'Description' column. The table shows a fault for 'Profile\_101' with the description 'Failed to form uni/infra/accportprof-Policy of class infraAccPortP'. The bottom of the interface shows a pagination bar with 'Page 1 Of 1', 'Objects Per Page: 15', and 'Displaying Objects 1 - 1 Of 1'.

In this case, correcting the fault would be as easy as creating a new Interface Selector Profile called 'Policy'.

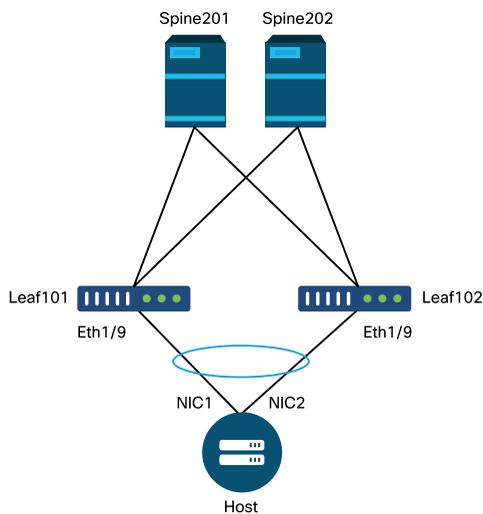
The manual configuration of basic access policies will be explored in the following paragraphs.

## Access policies manual basic configurations

When deploying access policies, objects are being defined to express the intended use of the given downlinks. The declaration which programs the downlinks (e.g. EPG Static Port assignment) relies on this expressed intent. This helps to scale the configuration and logically group similar use objects, such as switches or ports specifically connected to a given external device.

Reference the topology below for the remainder of this chapter.

### Topology of access policy definition for dual-homed server

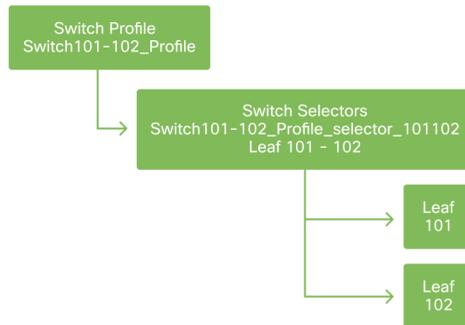


A web server is connected to an ACI fabric. The web server has 2 Network Interface Cards (NICs) which are configured in an LACP port-channel. The web server is connected to port 1/9 of leaf switches 101 and 102. The web server relies on VLAN-1501 and should reside in the EPG 'EPG-Web'.

## Configure the Switch Policy

The first logical step is to define which leaf switches will be used. The 'Switch Profile' will contain 'Switch Selectors' which define the leaf node IDs to be used.

### Switch policies



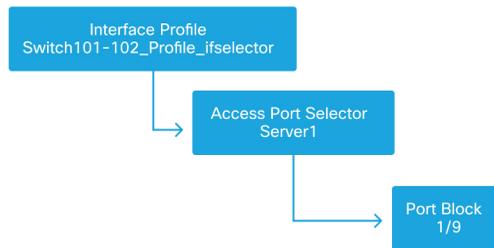
The general recommendation is to configure 1 Switch Profile per individual leaf switch and 1 Switch Profile per VPC domain pair, using a naming scheme which indicates the nodes which are part of the profile.

The Quick Start deploys a logical naming scheme which makes it easy to understand where it is applied. The completed name follows the 'Switch<node-id>\_Profile' format. As an example, 'Switch101\_Profile' will be for a switch profile containing leaf node 101 and Switch101-102\_Profile for a Switch Profile containing leaf nodes 101 and 102 which should be part of a VPC domain.

### Configure the Interface Policy

Once the switch access policies have been created, defining the interfaces would be the next logical step. This is done by creating an 'Interface Profile' which consists of 1 or more 'Access Port Selectors' which contain the 'Port Block' definitions.

#### Interface policies

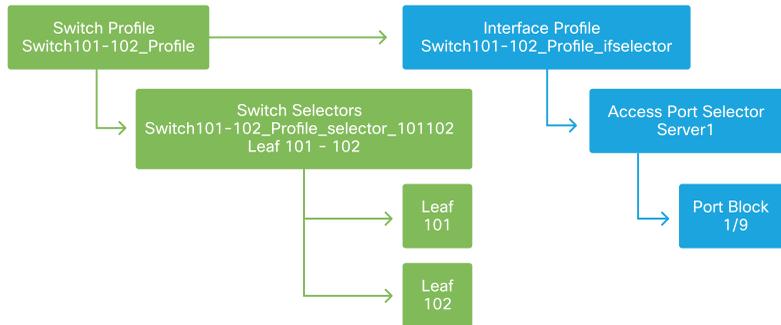


To form the relationship between the 'Interface Profile' and the switches involved, link the 'Switch Profile' to the 'Interface Profile'.

'Interface Profiles' can be defined in many ways. Similar to 'Switch Profiles', a single 'Interface Profile' can be created per physical switch along with an 'Interface Profile' per VPC domain. These policies should then have a 1-to-1 mapping to their corresponding switch profile. Following this logic, the fabric access policies are greatly simplified which makes it easy for other users to understand.

The default naming schemes employed by the Quick Start can also be used here. It follows the '<switch profile name>\_ifselector' format to indicate this profile is used to select interfaces. An example would be 'Switch101\_Profile\_ifselector'. This example 'Interface Profile' would be used to configure non VPC interfaces on leaf switch 101 and it would only be associated to the 'Switch101\_Profile' access policy.

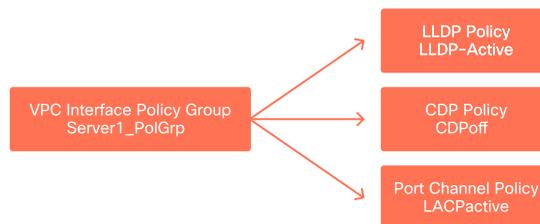
## Switch Profile associated to Interface Profile



Notice that since an 'Interface Profile' with Eth 1/9 is connected to a 'Switch Profile' which includes both leaf switch 101 and 102, provisioning Eth1/9 on both nodes begins simultaneously.

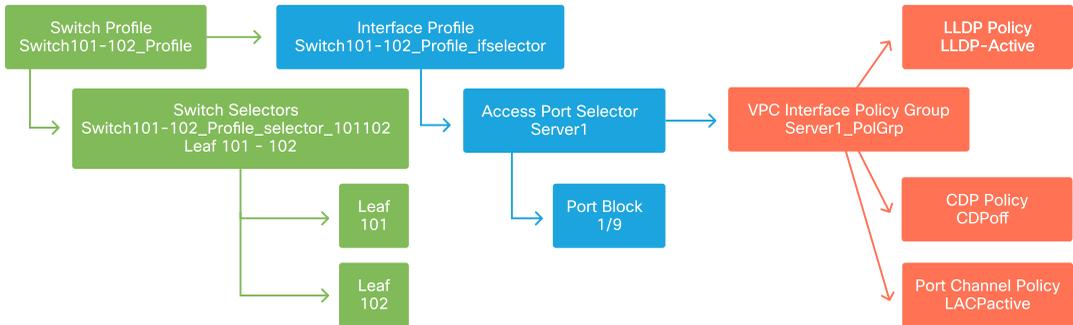
At this point, leaf switches and their ports have been defined. The next logical step would be to define characteristics of these ports. The 'Interface Policy Group' allows for the definition of these port properties. A 'VPC Interface Policy Group' will be created to allow for the above LACP Port-Channel.

## Policy Group



The 'VPC Interface Policy Group' gets associated to the 'Interface Policy Group' from the 'Access Port Selector' to form the relationship from leaf switch/Interface to port properties.

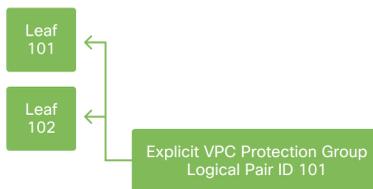
Switch and Interface Profiles combined



Configure the VPC

To create the LACP port-channel over 2 leaf switches, a VPC domain must be defined between leaf switch 101 and 102. This is done by defining a 'VPC Protection Group' between the two leaf switches.

VPC

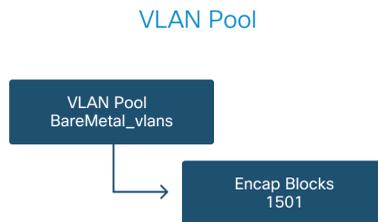


## Configure VLAN pools

The next logical step will be to create the VLANs that will be used on this port, in this case VLAN-1501. The definition of a 'VLAN Pool' with 'Encap Blocks' completes this configuration.

When considering the size of VLAN pool ranges, keep in mind that most deployments only need a single VLAN pool and one additional pool if using VMM integration. To bring VLANs from a legacy network into ACI, define the range of legacy VLANs as a static VLAN pool.

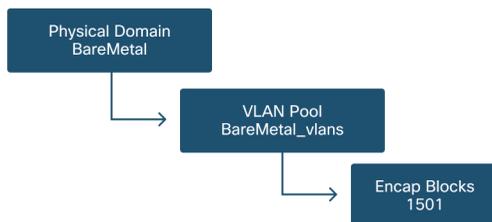
As an example, assume VLANs 1-2000 are used in a legacy environment. Create one Static VLAN pool which contains VLANs 1-2000. This will allow to trunk ACI Bridge Domains and EPGs towards the legacy fabric. If deploying VMM, a second dynamic pool can be created using a range of free VLAN IDs.



## Configure Domains

The next logical step is to create a 'Domain'. A 'Domain' defines the scope of a VLAN pool, i.e. where that pool will be applied. A 'Domain' could be physical, virtual, or external (bridged or routed). In this example, a 'Physical Domain' will be used to connect a bare metal server into the fabric. This 'Domain' gets associated to the 'VLAN Pool' to allow the required vlan(s).

## Physical Domains



For most deployments, a single 'Physical Domain' is enough for bare metal deployments and a single 'Routed Domain' is sufficient for L3Out deployments. Both can map to the same 'VLAN Pool'. If the fabric is deployed in a multi-tenancy fashion, or if more granular control is required to restrict which users can deploy specific EPGs & VLANs on a port, a more strategic access policy design should be considered.

'Domains' also provide the functionality to restrict user access to policy with 'Security Domains' using Roles Based Access Control (RBAC).

When deploying VLANs on a switch, ACI will encapsulate spanning-tree BPDUs with a unique VXLAN ID which is based on the domain the VLAN came from. Due to this, it is important to use the same domain whenever connecting devices which require STP communication with other bridges.

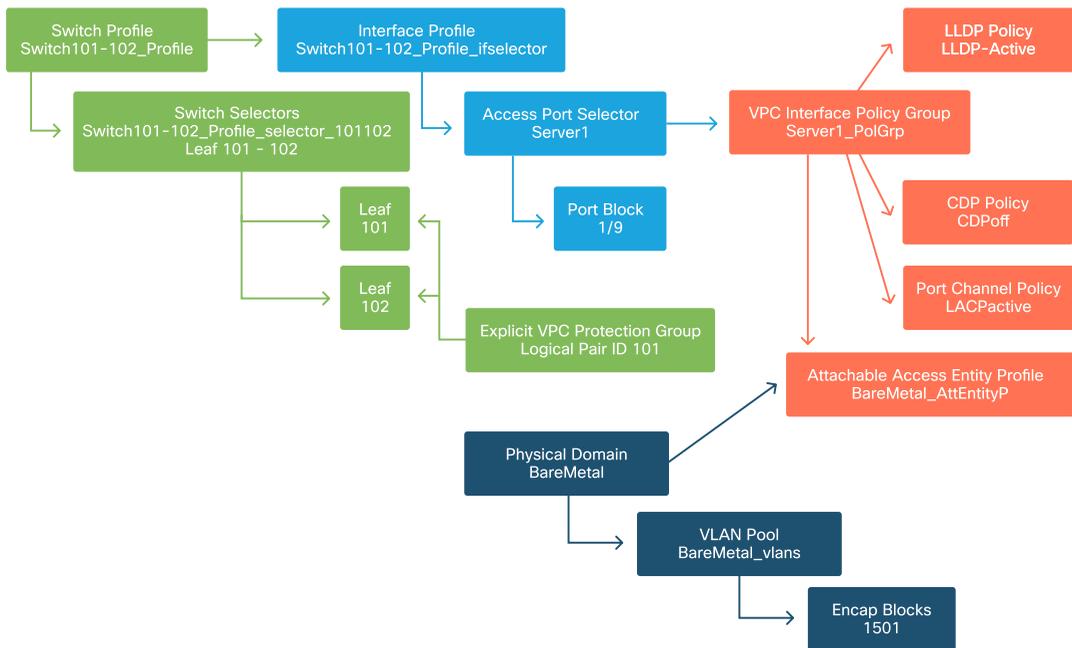
VLAN VXLAN IDs are also used to allow VPC switches to synchronize VPC learned MAC and IP addresses. Due to this, the simplest design for VLAN pools is to use a single pool for static deployments and create a second one for dynamic deployments.

### Configure the Attachable Access Entity Profile (AEP)

Two major chunks of access policy configuration have now been completed; the switch and interface definitions, and the domain/VLAN(s) definitions. An object called 'Attachable Access Entity Profile' (AEP) will serve to tie these two chunks together.

A 'policy group' is linked towards an AEP in a one-to-many relationship which allows for the AEP to group interfaces and switches together which share similar policy requirements. This means that only one AEP needs to be referenced when representing a group of interfaces on specific switches.

### Attachable Access Entity Profile



In most deployments, a single AEP should be used for static paths and one additional AEP per VMM domain.

The most important consideration is that VLANs can be deployed on interfaces through the AEP. This can be done by mapping EPGs to an AEP directly or by configuring a VMM domain for Pre-provision. Both these configurations make the associated interface a trunk port ('switchport mode trunk' on legacy switches).

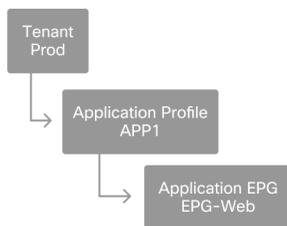
Due to this, it is important to create a separate AEP for L3Out when using routed ports or routed sub-interfaces. If SVIs are used in the L3Out, it is not necessary to create an additional AEP.

### Configure the tenant, APP, and EPG

ACI uses a different means of defining connectivity by using a policy-based approach.

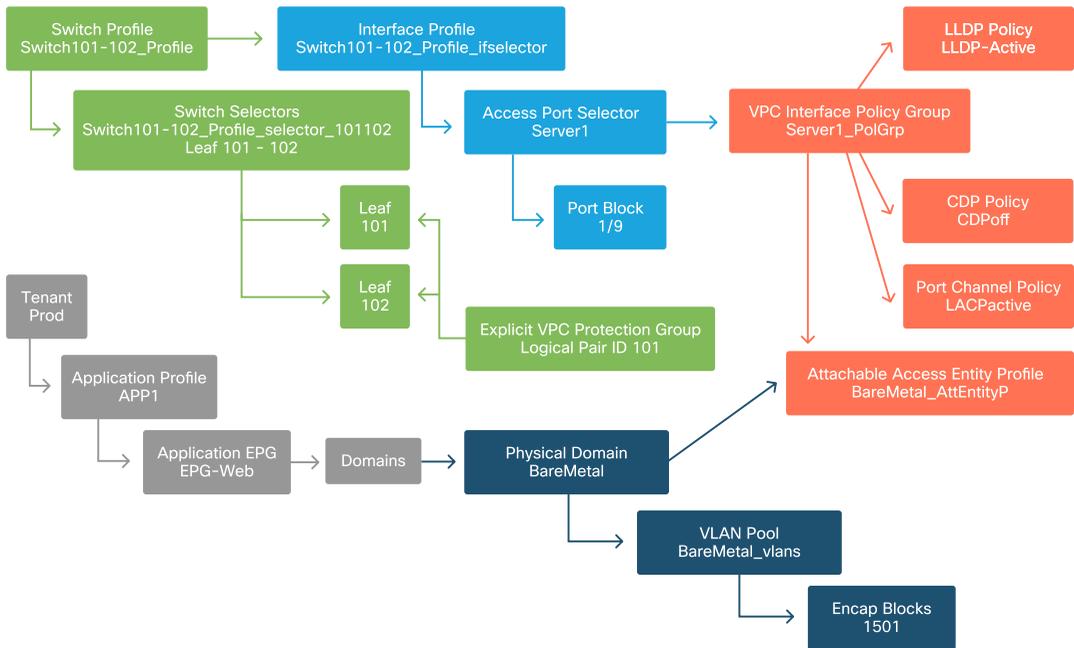
The lowest level object is called an 'Endpoint Group' (EPG). The EPG construct is used to define a group of VMs or servers (endpoints) with similar policy requirements. 'Application Profiles', which exist under a tenant, are used to logically group EPGs together.

#### Tenant, APP, and EPG



The next logical step is to link the EPG to the domain. This creates the link between the logical object representing our workload, the EPG, and the physical switches/interfaces, the access policies.

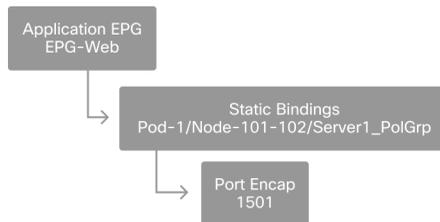
EPG to Domain link



### Configure the EPG Static Bindings

The last logical step is to program the VLAN onto a switch interface for a given EPG. This is especially important if using a physical domain, as this type of domain requires an explicit declaration to do so. This will allow the EPG to be extended out of the fabric and it will allow the bare metal server to get classified into the EPG.

#### Static Bindings

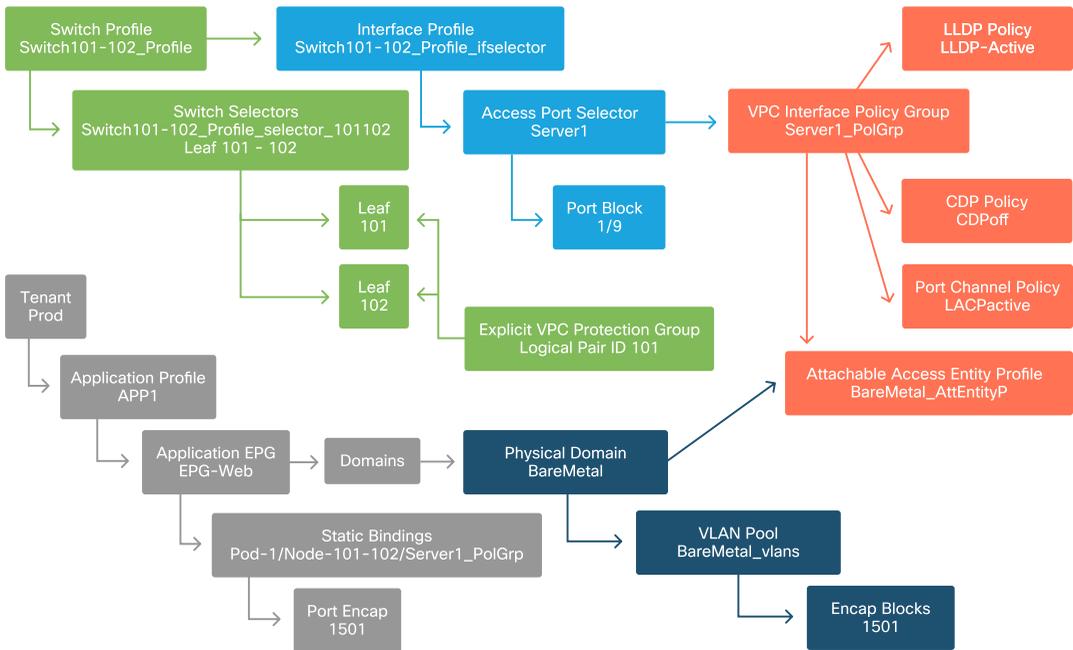


The referenced 'Port Encap' needs to be resolvable against the 'VLAN Pool'. If it is not, a fault will be flagged. This is discussed in the " Troubleshooting workflow" section of this chapter.

### Summary of the access policy configuration

The following diagram summarizes all the objects created to allow connectivity for the host through VLAN-1501, using a VPC connection to leaf switch 101 and 102.

Bare-metal ACI connectivity



### Connecting additional servers

With all the previous policies created, what would it mean to connect one more server on port Eth1/10 on leaf switches 101 and 102 with a port-channel?

Referring to the 'Bare-metal ACI connectivity' diagram, the minimum following will need to be created:

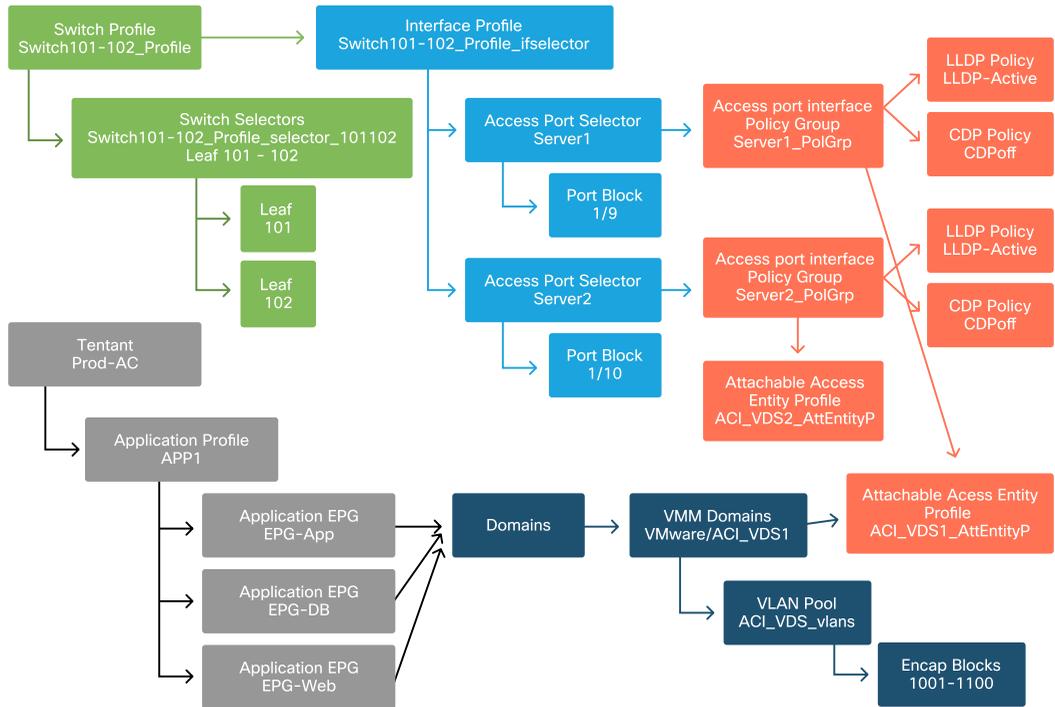
- An extra Access Port Selector and Port Block.
- An extra VPC Interface Policy Group.
- An extra Static Binding with Port Encap.

Note that for LACP port-channels, a dedicated VPC Interface Policy Group must be used as this VPC Policy Group is what defines the VPC id.

In the case of individual links, the non-VPC Interface Policy Group could be re-used for the extra server if the link requires the same port properties.

The resulting policies would look like the following image.

Connecting server2 into the setup



**What is next?**

The next section will go through a few access policy failure scenarios, starting with the topology and use case discussed in this overview.

## Troubleshooting workflow

The following troubleshooting scenarios could be encountered when working with access policies:

- A missing relationship between two or more entities in the access policy, such as access policy group not linked to an AEP.
- A missing or unexpected policy is tied to a given access policy, such as an LLDP policy named 'lldp\_enabled', while in reality the policy configuration has LLDP rx/tx disabled.
- A missing or unexpected value in the access policy, such as the configured VLAN ID encap missing from the configured VLAN Pool.
- A missing relationship between the EPG and access policy, such as no physical or virtual domain association to the EPG.

Most of the above troubleshooting involves walking through the access policy relationships to understand if any relationships are missing, or to understand which policies are configured and/or whether the configuration is resulting in the desired behavior.

### **Configure interface, PC, and VPC Quick Start**

Within the APIC GUI, the 'Configure Interface, PC, and VPC' quick start wizard facilitates access policy lookup by providing the administrator an aggregated view of existing access policies. This quick start wizard can be found in the GUI at:

'Fabric > Access Policies > Quick Start > Steps > Configure Interface, PC, and VPC'.

## Location of 'Configure Interface, PC, and VPC' Quick Start

The screenshot displays the Cisco APIC (Application Policy Infrastructure Controller) web interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Fabric' tab is selected, and the 'Access Policies' sub-tab is active. The left sidebar shows a tree view of 'Policies' with options like 'Quick Start', 'Switches', 'Modules', 'Interfaces', 'Policies', 'Pools', and 'Physical and External Domains'. The main content area is titled 'Quick Start' and is divided into three sections: 'Summary', 'Steps', and 'See Also'. The 'Steps' section lists several tasks, with 'Configure an interface, PC, and VPC' highlighted with a red box. The 'See Also' section lists related configuration tasks such as 'Physical Interface (Link Level)', 'CDP', 'LLDP', 'LACP', 'LACP Member', 'Spanning Tree Interface', 'Storm Control', 'Port Security', 'SPAN', 'On-demand Diagnostics', 'Attachable Entity Profile', 'QoS', and 'DHCP Relay'.

Even though the wizard has 'Configure' in the name, it is exceptionally handy for providing an aggregated view of the many access policies that must be configured to get interfaces programmed. This aggregation serves as a single view to understand which policies are already defined and effectively reduces the number of clicks required to begin isolating access policy-related issues.

## Using the Quick Start for troubleshooting

When the Quick Start view is loaded, the 'Configured Switch Interfaces' view (top-left pane) can be referenced to determine existing access policies. The wizard groups the

entries underneath folders that represent either individual or multiple leaf switches, depending on the access policies configuration.

As a demonstration of the wizard's value, the following wizard screenshots are presented, knowing the reader has no previous understanding of the fabric topology:

### Demo view of 'Configure Interface, PC, and VPC' Quick Start

#### Configure Interface, PC, and VPC

##### Configured Switch Interfaces

Switches	Interfaces	IF Type	Attached Device Type
101	1/31	Individ...	L3 (VLANs: 2600)
	1/4	Individ...	Bare Metal (VLANs: 311-3...
	1/25	Individ...	Bare Metal (VLANs: 1111,...
103-104	1/10	VPC	Bare Metal (VLANs: 100-3...
	1/6	VPC	Bare Metal (VLANs: 1590-...
	1/7	VPC	Bare Metal (VLANs: 1590-...
		VPC	Bare Metal (VLANs: 100-3...
	1/17	VPC	Bare Metal (VLANs: 700-7...
103	1/4	Individ...	L3 (VLANs: 3100,603,640,...
103,104			



##### VPC Switch Pairs

VPC Domain Id	Switch 1	Switch 2
34	103	104
58	105	108
67	107	106
212	2101	2102

The 'Configured Switch Interfaces' pane shows access policy mappings. The 'VPC Switch Pairs' pane shows completed VPC Protection Group definitions.

The table below shows a subset of completed access policy definitions that can be derived from the above screenshot.

Subset of completed access policies that can be derived from the above Quick Start view

Switch Node	Interface	Policy Group Type	Domain Type	VLANs
101	1/31	Individual	Routed (L3)	2600
101	1/4	Individual	Phys (Bare Metal)	311-3..?
103-104	1/10	VPC	Phys (Bare Metal)	100-3..?

The VLAN column entries are intentionally incomplete given the default view.

Similarly, the completed 'VPC Protection Group' policies can be derived from the 'VPC Switch Pairs' view (bottom-left pane). Without 'VPC Protection Groups', VPCs cannot be deployed as this is the policy which defines the VPC Domain between two leaf nodes.

Take into consideration that due to pane sizing, long entries are not completely visible. To view the full value of any entry, hover the mouse pointer on the field of interest.

Mouse pointer is hovering over 'Attached Device Type' field for 103-104, int 1/10 VPC entry:

### Configure Interface, PC, and VPC

Configured Switch Interfaces

Switches	Interfaces	IF Type	Attached Device Type
101	1/31	Individ...	L3 (VLANs: 2600)
	1/4	Individ...	Bare Metal (VLANs: 311-3...
	1/25	Individ...	Bare Metal (VLANs: 1111,...
103-104	1/10	VPC	Bare Metal (VLANs: 100-3...
	1/6	VPC	Bare Metal (VLANs: 1590...
	1/7	VPC	Bare Metal (VLANs: 1590...
		VPC	Bare Metal (VLANs: 100-3...
	1/17	VPC	Bare Metal (VLANs: 700-7...
103	1/4	Individ...	L3 (VLANs: 3100,603,640,...
103,104			



Click '+' to select switches or click table row to edit



Bare Metal (VLANs: 100-300,900-999), L3 (VLANs: 100-300,900-999)

VPC Switch Pairs

VPC Domain Id	Switch 1	Switch 2
34	103	104
58	105	108
67	107	106
212	2101	2102

By hovering the mouse over the pane, the complete entries are visible.

Updated subset of completed access policies using mouse-over details

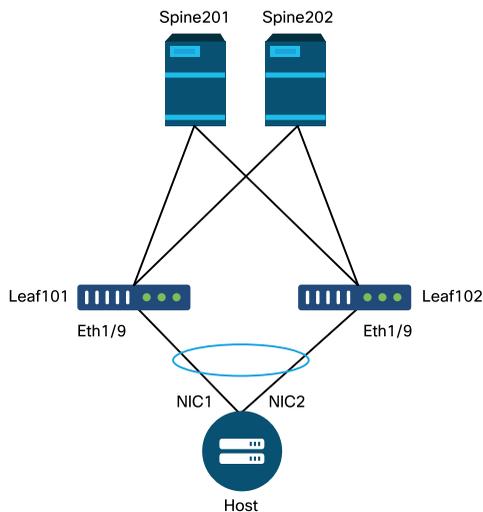
Switch Node	Interface	Policy Group Type	Domain Type	VLANs
101	1/31	Individual	Routed (L3)	2600
101	1/4	Individual	Phys (Bare Metal)	311-320
103-104	1/10	VPC	Phys (Bare Metal)	100-300,900-999
103-104	1/10	VPC	Routed (L3)	100-300,900-999

Full VLAN associations can now be observed and understood for troubleshooting and verification.

## Troubleshooting scenarios

For the following troubleshooting scenarios, reference the same topology from the previous chapter.

## Topology from access policy 'Introduction' section

**Scenario 1: Fault F0467 – invalid-path, nwissues**

This fault is raised when a switch/port/VLAN declaration is made without the corresponding access policies in place to allow that configuration to be applied properly. Depending on the description of this fault, a different element of the access policy relationship may be missing.

After deploying a static binding for the above VPC interface with trunked encap VLAN 1501 without the corresponding access policy relationship in place, the following fault is raised on the EPG:

**Fault:** F0467

**Description:** Fault delegate: Configuration failed for uni/tn-Prod1/ap-App1/epg-EPG-Web node 101\_101\_102\_eth1\_9 due to Invalid Path Configuration, Invalid VLAN Configuration, debug message: invalid-vlan: vlan-1501 :STP Segment Id not present for Encap. Either the EPG is not associated with a domain or the domain does not have this vlan assigned to it;invalid-path: vlan-1501 :There is no domain, associated with both EPG and Port, that has required VLAN;

From the above fault description, there are some clear indications as to what could be causing the fault to be triggered. There is a warning to check the access policy relationships, as well as to check the domain association to the EPG.

Reviewing the Quick Start view in the scenario described above, clearly the access policy is missing VLANs.

Quick Start view where 101-102, Int 1/9 VPC is missing VLANs

## Configure Interface, PC, and VPC

### Configured Switch Interfaces

Switches	Interfaces	IF Type	Attached Device Type
101-102	1/11	Individual	ESX (VLANs: 1001-1100)
101-102	1/9	VPC	Bare Metal
101	1/17	Individual	L3 (VLANs: 901-910)
102	1/19	Individual	L3 (VLANs: 901-910)
301-302	1/11	Individual	ESX (VLANs: 1001-1100)
301	1/17	Individual	L3 (VLANs: 901-910)
302	1/19	Individual	L3 (VLANs: 901-910)



Click '+' to select switches or click table row to edit



### VPC Switch Pairs

VPC Domain Id	Switch 1	Switch 2
101	101	102

Note that the entry is missing a reference to any VLAN IDs.

Once corrected, the Quick Start view will show '(VLANs 1500-1510)':

101-102, Int 1/9 VPC now shows Bare Metal (VLANs: 1500-1510)

## Configure Interface, PC, and VPC

Configured Switch Interfaces

Switches	Interfaces	IF Type	Attached Device Type
101-1...	1/11	Individual	ESX (VLANs: 1001-1100)
	1/9	VPC	Bare Metal (VLANs: 1500-1510)
101	1/17	Individual	L3 (VLANs: 901-910)
102	1/19	Individual	L3 (VLANs: 901-910)
301-3...	1/11	Individual	ESX (VLANs: 1001-1100)
301	1/17	Individual	L3 (VLANs: 901-910)
302	1/19	Individual	L3 (VLANs: 901-910)



Click '+' to select switches or click table row to edit



Bare Metal (VLANs: 1500-1510)

VPC Switch Pairs

VPC Domain Id	Switch 1	Switch 2
101	101	102

However, the EPG fault still exists with the following updated description for fault F0467:

### Fault: F0467

**Description:** Fault delegate: Configuration failed for uni/tn-Prod1/ap-App1/epg-EPG-Web node 101 101\_102\_eth1\_9 due to Invalid Path Configuration, debug message: invalid-path: vlan-150 : There is no domain, associated with both EPG and Port, that has required VLAN.

With the above updated fault, check the EPG domain associations to find that there are no domains tied to the EPG.

EPG-Web has Static Ports association, but is missing domain associations

The screenshot shows the Cisco APIC interface for the 'Prod1' tenant. The 'Tenants' tab is selected, and the 'Prod1' tenant is active. The left-hand navigation pane shows the configuration tree, with 'Application Profiles' > 'App1' > 'Application EPGs' > 'EPG-Web' highlighted. The 'Domains (VMs and Bare-Metals)' table is currently empty, displaying the message: 'No items have been found. Select Actions to create a new item.'

Dom Type	Deploy	Resolu	Allow Micro-Segme	Primary VLAN	Port Encap	Switch Mode	Encap Mode	Cos Value	Enhance Lag Policy
No items have been found. Select Actions to create a new item.									

Once the domain that contains VLAN 1501 is associated to the EPG, no further faults are raised.

### Scenario 2: Unable to select VPC as a path to deploy on EPG Static Port or L3Out Logical Interface Profile (SVI)

While attempting to configure a VPC as a path on an EPG Static Port or L3Out Logical Interface Profile SVI entry, the specific VPC to be deployed is not displayed as an available option.

When attempting to deploy a VPC static binding, there are two hard requirements:

- 1 The VPC Explicit Protection Group must be defined for the pair of leaf switches in question.
- 2 The full access policy mapping must be defined.

Both requirements can be checked from the Quick Start view as shown above. If neither is complete, the VPC will simply not show up as an available option for Static Port Bindings.

### Scenario 3: Fault F0467 – fabric encap already used in another EPG

By default, VLANs have a global scope. This means that a given VLAN ID can only be used for a single EPG on a given leaf switch. Any attempt to re-use the same VLAN on multiple EPGs within a given leaf switch will result in the following fault:

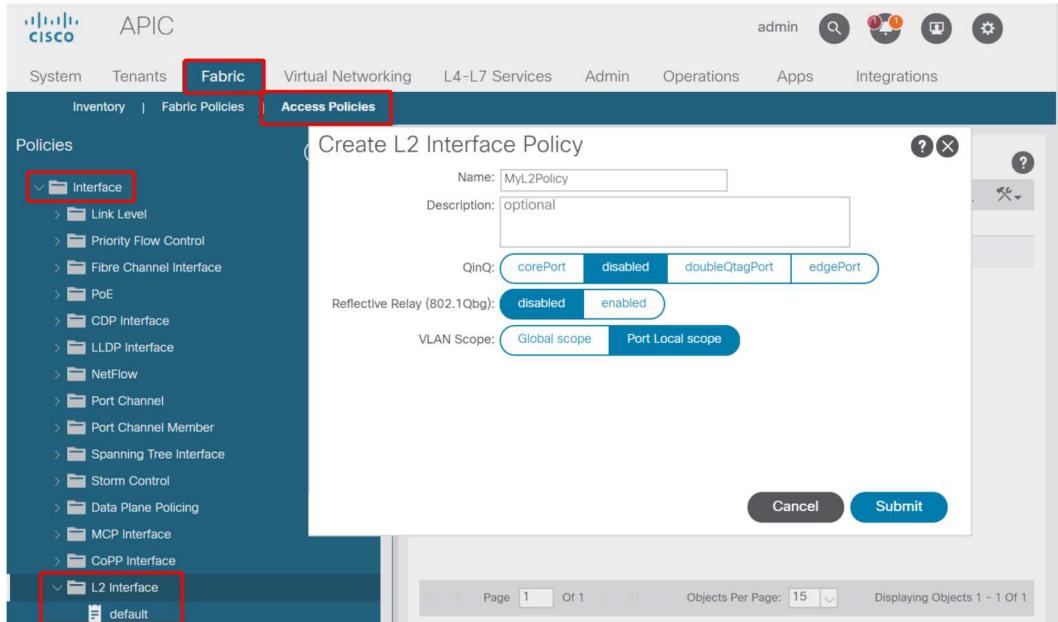
#### **Fault: F0467**

**Description:** Fault delegate: Configuration failed for uni/tn-Prod1/ap-App1/epg-EPG-BusinessApp node 102 101\_102\_eth1\_8 due to Encap Already Used in Another EPG, debug message: encap-already-in-use: Encap is already in use by Prod1:App1:EPG-Web;

Aside from selecting a different VLAN, another option to make this configuration work is to consider the usage of 'Port Local' VLAN Scope. This scope allows for VLANs to be mapped on a per-interface basis which means that VLAN-1501 could potentially be used for different EPGs, across multiple interfaces, on the same leaf.

Although 'Port Local' scope gets associated on a Policy Group basis (specifically via an L2 policy), it is applied at the leaf level.

Location to change 'VLAN Scope' setting within APIC GUI



Before implementing the 'Port Local' VLAN scope configuration, review the "Cisco APIC Layer 2 Networking Configuration Guide" on Cisco.com to ensure that its limitations and design restrictions are acceptable for the desired use cases and designs.

## Special mentions

### Show Usage

While not specific to access policies, a button is available on most objects in the GUI that is labeled 'Show Usage'. This button performs a policy lookup rooted at the selected object to determine which leaf nodes/interfaces have a direct relationship to it. This can be useful for both the general lookup scenario as well as to gain an understanding of whether a specific object or policy is even in use.

In the screenshot below, the selected AEP is being used by two different interfaces. This implies that making a modification to the AEP will have a direct impact on the associated interfaces.

'Show Usage' view when used on 'Attachable Access Entity Profile'

The screenshot shows the Cisco APIC interface. The left sidebar has a navigation menu with 'Fabric' and 'Access Policies' highlighted. The main content area displays the 'Policy Usage Information' view for an Attachable Access Entity Profile (AEP). The view includes a title bar, a navigation menu, and a main content area with a table of nodes using the policy and a table of policies using the policy.

**Nodes using this policy**

Node Id	Name	Resources
101	STP1-Leaf101	<a href="#">Click to Show Detail</a>
102	STP1-Leaf102	<a href="#">Click to Show Detail</a>

**Policies using this policy**

Name	Type
101_102_eth1_9	PG/PC Interface Policy Group

Buttons: Change Deployment Settings, Close, Show Usage, Reset, Submit

## Overlapping VLAN Pools

While the function of access policies is to allow a specific VLAN to be deployed onto an interface, there is additional usage that must be considered during the design phase. Specifically, the domain gets used in the calculation of the VXLAN ID (called Fabric Encap) tied to the external encapsulation. While this functionality generally has no major bearing on dataplane traffic, such IDs are especially relevant for a subset of protocols which flood through the fabric, including Spanning Tree BPDUs. If VLAN-*<id>* BPDUs ingressing on leaf1 are expected to egress Leaf 2 (e.g. having legacy switches converging spanning-tree through ACI), VLAN-*<id>* must have the same fabric encap on both leaf nodes. If the fabric encap value differs for the same access VLANs, the BPDUs will not traverse the fabric.

As mentioned in the previous section, avoid configuration of same VLANs in multiple domains (VMM vs Physical, for example) unless special care is being taken to ensure

that each domain is only ever applied to a unique set of leaf switches. The moment both domains can be resolved onto the same leaf switch for a given VLAN, there is a chance that underlying VXLAN can be changed after an upgrade (or clean reload) which can lead for example to STP convergence issues. The behavior is a result of each domain having a unique numerical value (the 'base' attribute) which is used in the following equation to determine VXLAN ID:

$$\text{VXLAN VNID} = \text{Base} + (\text{encap} - \text{from\_encap})$$

To validate which domains are pushed onto a given leaf, a moquery can be run against the 'stpAllocEncapBlkDef' class:

```
leaf# moquery -c stpAllocEncapBlkDef
# stp.AllocEncapBlkDef
encapBlk      : uni/infra/vlanns-[physvlans]-dynamic/from-[vlan-1500]-to-[vlan-1510]
base         : 8492
dn           : allocencap-[uni/infra]/encapsdef-[uni/infra/vlanns-[physvlans]-dynamic]/allocencapblkdef-[uni/infra/vlanns-[physvlans]-dynamic/from-[vlan-1500]-to-[vlan-1510]]
from         : vlan-1500
to           : vlan-1510
```

From this output, discern the following access policy definitions:

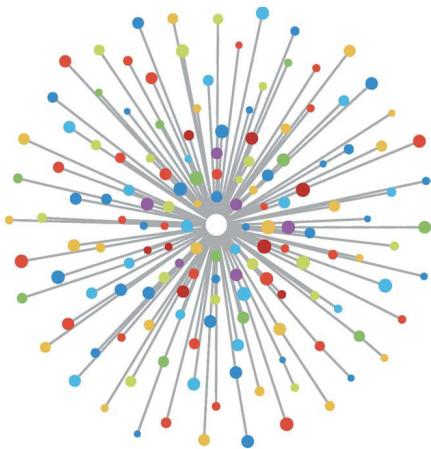
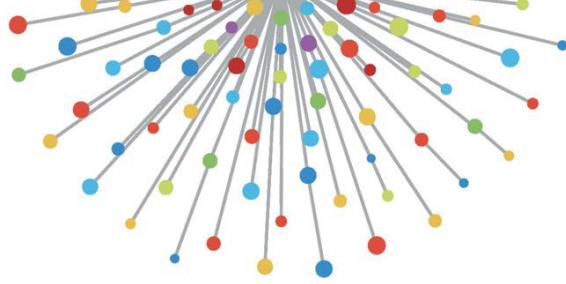
- There is a programmed VLAN pool with a block of VLANs explicitly defining VLANs 1500-1510.
- This block of VLANs is tied to a domain named 'physvlans'.
- The base value used in VXLAN calculation is 8492.
- The resulting VXLAN calculation for VLAN-1501 would be  $8492 + (1501-1500) = 8493$  as the fabric encapsulation.

The resulting VXLAN ID (in this example, 8493) can be verified with the following command:

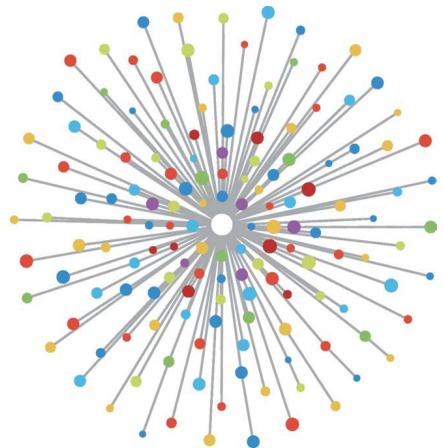
```
leaf# show system internal epm vlan all
```

VLAN ID	Type	Access Encap (Type Value)	Fabric Encap	H/W id	BD VLAN	Endpoint Count
13	Tenant BD	NONE	0 16121790	18	13	0
<b>14</b>	<b>FD vlan 802.1Q</b>		<b>1501 8493</b>	<b>19</b>	<b>13</b>	<b>0</b>

If there is any other VLAN pool containing VLAN-1501 that gets pushed onto the same leaf, an upgrade or clean reload could potentially grab unique base value (and subsequently a different Fabric Encap) which will cause BPDUs to stop making it to another leaf which is expected to receive BPDUs on VLAN-1501.



# Security policies



## Overview

The fundamental security architecture of the ACI solution follows a whitelist model. Unless a VRF is configured in **unenforced** mode, all EPG to EPG traffic flows are implicitly dropped. As implied by the out-of-the-box whitelist model, the default VRF setting is in **enforced** mode. Traffic flows can be allowed or explicitly denied by implementing zoning-rules on the switch nodes. These zoning-rules can be programmed in a variety of different configurations depending on the desired communication flow between endpoint groups (EPG) and the method used to define them. Note that zoning-rule entries are not stateful and will typically allow/deny based on port/socket given two EPGs once the rule has been programmed.

### Methods to program zoning-rules

The main methods to program zoning-rules within ACI are as follows:

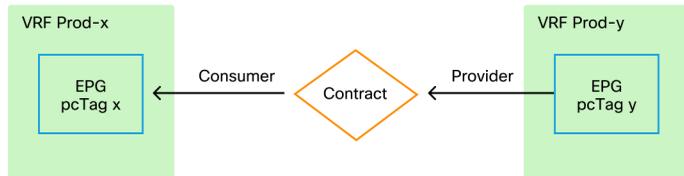
- **EPG-to-EPG Contracts:** Typically requires at least one consumer and one provider to program zoning-rules across two or more distinct endpoint groups.
- **Preferred Groups:** Requires enabling grouping at the VRF level; only one group can exist per VRF. All members of the group can communicate freely. Non-members require contracts to allow flows to the preferred group.
- **vzAny:** An 'EPG Collection' that is defined under a given VRF. vzAny represents all EPGs in the VRF. Usage of vzAny allows flows between one EPG and all EPGs within the VRF via one contract connection.

The following diagram can be used to reference the granularity of zoning-rule that each of the above methods allows for control:

## Comparison between zoning-rule methodologies

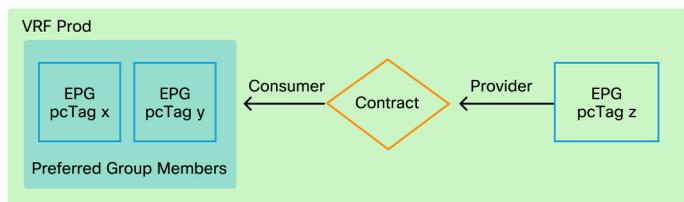
### Contract

- EPG to EPG granularity
- Requires at least 1 consumer and 1 provider
- Can scope across VRFs/Tenants



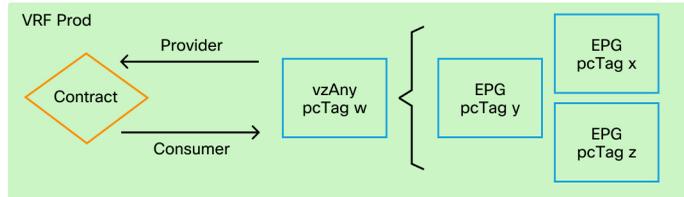
### Preferred Groups

- Must be enabled per VRF
- Only one group per VRF
- EPGs must be explicitly added
- All members communicate freely
- Non-Members require contracts to communicate with members



### vzAny

- Exists within a VRF
- Requires contracts to allow flows
- Zoning-rules apply to all EPGs within the VRF



While utilizing the contract method of programming zoning-rules, there is an option for defining the contract scope. This option must be given careful consideration if any route leaking/shared service design is required. If the wish is to get from one VRF to another within the ACI fabric, contracts are the method to do so.

The scope values can be the following:

- **Application:** a contract consumer/provider relationship will only program rules between EPGs that are defined within the same Application Profile. Re-using the same contract across other Application Profile EPGs will not allow for crosstalk between them.

- **VRF (default):** a contract consumer/provider relationship will program rules between EPGs that are defined within the same VRF. Re-using the same contract across other Application Profile EPGs will allow for crosstalk between them. Take care to ensure that only desired flows are allowed, otherwise a new contract should be defined to prevent unintentional crosstalk.
- **Tenant:** a contract consumer/provider relationship will program rules between EPGs that are defined within the same tenant. If there are EPGs tied to multiple VRFs within a single tenant and they consume/provide the same contract, this scope can be used to induce route leaking to allow for inter-VRF communication.
- **Global:** a contract consumer/provider relationship will program rules between EPGs across any tenant within an ACI fabric. This is the highest possible scope of the definition, and great care should be taken when this is enabled on previously defined contracts as to prevent unintentional flow leakage.

## Reading a zoning-rule entry

Once the zoning-rule is programmed, it will appear as the following on a leaf:

Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name	Action	Priority
---------	--------	--------	----------	-----	--------	-------	------	--------	----------

- **Rule ID:** the ID of the rule entry. No real significance other than to act as a unique identifier.
- **Src EPG:** a unique ID per VRF (pcTag) of the source endpoint group.
- **Dst EPG:** a unique ID per VRF (pcTag) of the destination endpoint group.
- **FilterID:** the ID of the filter that the rule is attempting to match against. The Filter contains the protocol info that the rule will match against.
- **Dir:** the directionality of the zoning-rule.

- **OperSt:** the operating State of the rule.
- **Scope:** a unique ID of the VRF that the rule will match against.
- **Name:** the name of the contract that resulted in that entry being programmed.
- **Action:** what the leaf will do when it matches that entry. Includes: [Drop, Permit, Log, Redirect].
- **Priority:** the order in which the zoning-rules will be validated for action given a matching Scope, SrcEPG, DstEPG, and Filter Entries.

## Policy Content-Addressable Memory (CAM)

As each zoning rule gets programmed, a matrix of the zoning-rule entry mapped against filter entries will begin to consume **Policy CAM** on the switches. While designing allowed flows through an ACI fabric, special care should be taken when re-using contracts, as opposed to creating new ones, depending on the end design. Haphazardly re-using the same contract across multiple EPGs without understanding the resulting zoning-rules can quickly cascade into multiple flows being allowed unexpectedly. At the same time, these unintentional flows will continue to consume Policy CAM. When Policy CAM becomes full, the zoning-rule programming will begin to fail which can result in unexpected and intermittent loss depending on configuration and endpoint behaviors.

## VRF leaking, global pcTags and policy enforcement directionality of shared L3Outs

This is a special callout for the shared services use case which requires contracts to be configured. Shared services typically imply inter-VRF traffic within an ACI fabric which relies on the usage of either a 'tenant' or 'global' scoped contract. To fully understand this, one must first reinforce the idea that the typical pcTag value assigned to EPGs are not globally unique. pcTags are scoped to a VRF and the same pcTag could potentially be re-used within another VRF. When the discussion of route leaking comes up, start to

enforce requirements on the ACI fabric including the need for globally unique values including subnets and pcTags.

What makes this a special consideration is the directionality aspect tied to an EPG being a consumer vs a provider. In a shared services scenario, the provider is typically expected to drive a global pcTag to get a fabric unique value. At the same time, the consumer will retain its VRF-scoped pcTag which puts it in a special position to be able to now program and understand the usage of the global pcTag value to enforce policy.

For reference, the pcTag allocation range is as follows:

- System reserved: 1-15.
- Global scoped: 16-16384 for shared services provider EPGs.
- Local scoped: 16385-65535 for VRF scoped EPGs.

## **VRF policy control enforcement direction**

In each VRF it is possible to define the enforcement direction setting.

- The default setting of enforcement direction is Ingress.
- The other option for enforcement direction is Egress.

Understanding where the policy is enforced depends on several different variables.

The table below helps to understand where the security policy is enforced at leaf level.

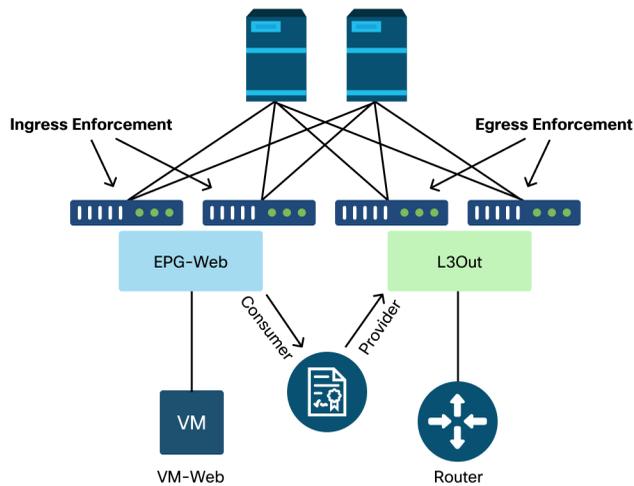
## Where is policy enforced?

Scenario	VRF enforcement mode	Consumer	Provider	Policy enforced on
Intra-VRF	Ingress/egress	EPG	EPG	<ul style="list-style-type: none"> <li>If destination endpoint is learned: ingress leaf*</li> <li>If destination endpoint is not learned: egress leaf</li> </ul>
	Ingress	EPG	L3Out EPG	Consumer leaf (non-border leaf)
	Ingress	L3Out EPG	EPG	Provider leaf (non-border leaf)
	Egress	EPG	L3Out EPG	Border leaf -> non-border leaf traffic <ul style="list-style-type: none"> <li>If destination endpoint is learned: border leaf</li> <li>If destination endpoint is not learned: non-border leaf</li> </ul>
	Egress	L3Out EPG	EPG	Non-border leaf-> border leaf traffic <ul style="list-style-type: none"> <li>Border leaf</li> </ul>
	Ingress/egress	L3Out EPG	L3Out EPG	Ingress leaf*
Inter-VRF	Ingress/egress	EPG	EPG	Consumer leaf
	Ingress/egress	EPG	L3Out EPG	Consumer leaf (Non-border leaf)
	Ingress/egress	L3Out EPG	EPG	Ingress leaf*
	Ingress/egress	L3Out EPG	L3Out EPG	Ingress leaf*

\*Policy enforcement is applied on the first leaf hit by the packet.

The figure below illustrates an example of contract enforcement where EPG-Web as consumer and L3Out EPG as provider have an intra-VRF contract. If VRF is set to Ingress enforcement mode, policy is enforced by the leaf nodes where EPG-Web resides. If VRF is set to Egress enforcement mode, policy is enforced by the border leaf nodes where L3Out resides if VM-Web endpoint is learned on the border leaf.

#### Ingress enforcement and egress enforcement



# Tools

There are a variety of tools and commands that can be used to help in the identification of a **policy drop**. A policy drop can be defined as a packet drop due to a contract configuration or lack thereof.

## Zoning-rule validation

The following tools and commands can be used to explicitly validate the zoning-rules that are programmed on leaf switches as a result of completed contract consumer/provider relationships.

### 'show zoning-rules'

A switch level command showing all zoning rules in place.

```
leaf# show zoning-rule
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir | operSt | Scope | Name | Action | Priority |
|-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 4156 | 25 | 16410 | 425 | uni-dir-ignore | enabled | 2818048 | external_to_ntp | permit | fully_qual(7) |
| 4131 | 16410 | 25 | 424 | bi-dir | enabled | 2818048 | external_to_ntp | permit | fully_qual(7) |
|-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

### 'show zoning-filter'

A filter that contains the sport/dport information that the zoning rule is acting on. The filter programming can be verified with this command.

```
leaf# show zoning-filter
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| FilterId | Name | EtherT | Prot | ApplyToFrag | Stateful | SFromPort | SToPort | DFromPort | DToPort | Prio |
|-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| implarp | implarp | arp | unspecified | no | no | unspecified | unspecified | unspecified | unspecified | dport |
| implicit | implicit | unspecified | unspecified | no | no | unspecified | unspecified | unspecified | unspecified | implicit |
| 425 | 425_0 | ip | tcp | no | no | 123 | 123 | unspecified | unspecified | sport |
| 424 | 424_0 | ip | tcp | no | no | unspecified | unspecified | 123 | 123 | dport |
|-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

**'show system internal policy-mgr stats'**

This command can be run to verify the number of hits per zoning-rule. This is useful to determine whether an expected rule is being hit as opposed to another, such as an implicit drop rule that may have a higher priority.

```
leaf# show system internal policy-mgr stats
Requested Rule Statistics
Rule (4131) DN (sys/actrl/scope-2818048/rule-2818048-s-16410-d-25-f-424) Ingress: 0, Egress: 0, Pkts: 0 RevPkts: 0
Rule (4156) DN (sys/actrl/scope-2818048/rule-2818048-s-25-d-16410-f-425) Ingress: 0, Egress: 0, Pkts: 0 RevPkts: 0
```

**'show logging ip access-list internal packet-log deny'**

A switch level command that can be run at iBash level which reports ACL (contract) related drops and flow-related information including:

- VRF
- VLAN-ID
- Source MAC/Dest MAC
- Source IP/Dest IP
- Source Port/Dest Port
- Source Interface

```
leaf# show logging ip access-list internal packet-log deny
[ Tue Oct 1 10:34:37 2019 377572 usecs]: CName: Prod1:VRF1(VXLAN: 2654209), VlanType: Unknown, Vlan-Id: 0, SMac: 0x000c0c0c0c0c,
DMac:0x000c0c0c0c0c, SIP: 192.168.21.11, DIP: 192.168.22.11, SPort: 0, DPort: 0, Src Intf: Tunnel7, Proto: 1, PktLen: 98
[ Tue Oct 1 10:34:36 2019 377731 usecs]: CName: Prod1:VRF1(VXLAN: 2654209), VlanType: Unknown, Vlan-Id: 0, SMac: 0x000c0c0c0c0c,
DMac:0x000c0c0c0c0c, SIP: 192.168.21.11, DIP: 192.168.22.11, SPort: 0, DPort: 0, Src Intf: Tunnel7, Proto: 1, PktLen: 98
```

**contract\_parser**

An on-device Python script which produces an output that correlates the zoning-rules, filters and hit statistics while performing name lookups from IDs. This script is extremely useful in that it takes a multi-step process and turns it into a single

command which can be filtered to specific EPGs/VRFs or on other contract related values.

```
leaf# contract_parser.py
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-l4] dst-epg [dst-l4] [flags][contract:{str}] [hit-count]

[7:4131] [vrf:common:default] permit ip tcp tn-Prod1/ap-Services/epg-NTP(16410) tn-Prod1/13out-L30ut1/instP-extEpg(25) eq 123
[contract:uni/tn-Prod1/brc-external_to_ntp] [hit=0]
[7:4156] [vrf:common:default] permit ip tcp tn-Prod1/13out-L30ut1/instP-extEpg(25) eq 123 tn-Prod1/ap-Services/epg-NTP(16410)
[contract:uni/tn-Prod1/brc-external_to_ntp] [hit=0]
[12:4169] [vrf:common:default] deny,log any tn-Prod1/13out-L30ut1/instP-extEpg(25) epg:any [contract:implicit] [hit=0]
[16:4167] [vrf:common:default] permit any epg:any tn-Prod1/bd-Services(32789) [contract:implicit] [hit=0]
```

## Packet classification validation

### ELAM

An ASIC level report used to check forwarding details which indicates, in the case of a dropped packet, the drop reason. Relevant to this section, the reason can be a SECURITY\_GROUP\_DENY (contract policy drop).

### fTriage

A Python-based utility on the APIC which can track end-to-end packet flow with ELAM.

### ELAM Assistant App

An APIC App that abstracts the complexity of various ASICs to make forwarding decision inspection much more convenient and user friendly.

Please refer to the "Intra-Fabric Forwarding" section for additional details on the ELAM, fTriage and ELAM Assistant Tools

## Policy CAM usage

Policy CAM usage on a per leaf basis is an important parameter to monitor to ensure the fabric is in a healthy status. The quickest way to monitor that is to use the 'Capacity Dashboard' within the GUI and explicitly check the 'Policy Cam' column.

The 'Leaf Capacity' view of Capacity Dashboard

The screenshot shows the Cisco APIC Capacity Dashboard in the 'Leaf Capacity' view. The dashboard includes a navigation bar with 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. Below the navigation bar, there are tabs for 'Visibility & Troubleshooting', 'Capacity Dashboard', 'EP Tracker', and 'Visualization'. The main content area displays a table of capacity metrics for various pods. The table has columns for Switch, VRF, BD, EPG, Mac (learned), IPv4 (learned), IPv6 (learned), Multicast, and Policy CAM. The Policy CAM column shows rules and labels for each pod.

Switch	VRF	BD	EPG	Mac (learned)	IPv4 (learned)	IPv6 (learned)	Multicast	Policy CAM
pod-1/node-101 N9K-C93180YC-FX Configure Profile	<1% 4 of 800	<1% 2 of 3500	<1% 1 of 3960	<1% 3 of 24576 Local: 3 Remote: 0	<1% 2 of 24576 Local: 2 Remote: 0	0% 0 of 12288 Local: 0 Remote: 0	0% 0 of 8192	Rules: 44 of 65536 Labels: 0
pod-1/node-102 N9K-C93180YC-FX Configure Profile	<1% 4 of 800	<1% 2 of 3500	<1% 1 of 3960	<1% 4 of 24576 Local: 4 Remote: 0	<1% 1 of 24576 Local: 1 Remote: 0	0% 0 of 12288 Local: 0 Remote: 0	0% 0 of 8192	Rules: 40 of 65536 Labels: 0
pod-2/node-301 N9K-C93180YC-FX Configure Profile	<1% 3 of 800	<1% 2 of 3500	<1% 1 of 3960	<1% 3 of 24576 Local: 3 Remote: 0	<1% 1 of 24576 Local: 1 Remote: 0	0% 0 of 12288 Local: 0 Remote: 0	0% 0 of 8192	Rules: 38 of 65536 Labels: 0
pod-2/node-302 N9K-C93180YC-FX Configure Profile	<1% 3 of 800	<1% 2 of 3500	<1% 1 of 3960	<1% 3 of 24576 Local: 3 Remote: 0	<1% 2 of 24576 Local: 2 Remote: 0	0% 0 of 12288 Local: 0 Remote: 0	0% 0 of 8192	Rules: 42 of 65536 Labels: 0

### 'show platform internal hal health-stats'

This command is useful for validating a variety of resource limits and usage, including Policy CAM. Note that this command can only be run in vsh\_lc, so pass it in using the '-c' flag if being run from iBash.

```
leaf8# vsh_lc -c "show platform internal hal health-stats"
|Sandbox_ID: 0 Asic Bitmap: 0x0
|-----
...

Policy stats:
=====
policy_count           : 96
max_policy_count      : 65536
policy_otcam_count    : 175
max_policy_otcam_count : 8192
policy_label_count    : 0
max_policy_label_count : 0
=====
```

# EPG to EPG

## Generic policy drop considerations

There are numerous ways to troubleshoot a connectivity issue between two endpoints. The following methodology provides a good starting point to quickly and effectively isolate whether the connectivity issue is the result of a **policy drop** (contract induced).

Some high-level questions worth asking before diving in:

- Are the endpoints in same or different EPG?
  - Traffic between two endpoints residing in different EPGs (inter-EPG) is implicitly denied and requires a contract to allow communication.
  - Traffic between two endpoints within the same EPG (intra-EPG) is implicitly allowed, unless intra-EPG isolation is in use.
- Is the VRF enforced or unenforced?
  - When a VRF is in **enforced** mode, – within the VRF – contracts are required for endpoints in two different EPGs to communicate.
  - When a VRF is in **unenforced** mode, – within the VRF – all traffic would be allowed by the ACI fabric across multiple EPGs belonging to the unenforced VRF, regardless of the ACI contracts applied.

## Methodology

With the various tools available, there are some that are more appropriate and convenient to start with than others, depending on the level of information already known about the affected flow.

Is the full path of the packet in the ACI fabric known (ingress leaf, egress leaf...)?

If the answer is yes, ELAM Assistant should be used to identify the drop reason on the source or destination switch.

If the answer is no, Visibility & Troubleshooting, fTriage, contract\_parser, Operational tab in the Tenant view, and iBash commands will help to narrow down the path of the packet or give more visibility into the drop reasons.

Please note that the fTriage tool will not be discussed in detail in this section. Refer to the chapter "Intra-Fabric Forwarding" for more detail on using this tool.

Consider that while Visibility & Troubleshooting can help to quickly visualize where packets are dropped between two endpoints, fTriage shows more in-depth information for further troubleshooting. i.e. fTriage will help identify interface, drop reason, and other low-level details about the affected flow.

This example scenario will show how to troubleshoot a policy drop between two endpoints: 192.168.21.11 and 192.168.23.11

Assuming packet drops are experienced between those two endpoints, the following troubleshooting workflow will be used to identify the root cause of the problem:

Identify the src/dst leaf(s) involved in the traffic flow:

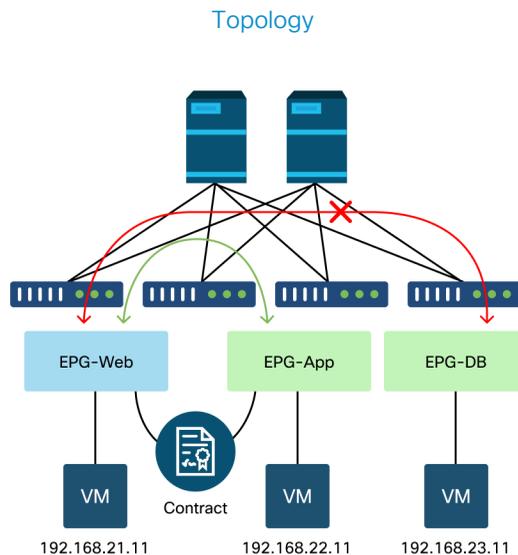
- 1 Use **Visibility & Troubleshooting** to trace the packet flow and identify which device is dropping the packet.
- 2 Run the command 'show logging ip access-list internal packet-log deny' on the selected device.
  - If a packet with one of the IP addresses of interest is being denied and logged, the **packet-log** will print the relevant endpoint and contract name on a per hit basis.

- 3 Use command 'contract\_parser.py --vrf <tenant>:<VRF>' on source and destination leaf to observe hit count for the configured contract:
  - If a packet is hitting the contract on either the source or destination switch, the counter of the relevant contract will increment
  - This method is less granular than that of IP access-list internal packet-log in situations where many flows could be hitting the same rule (many endpoints/flows between the two EPGs of interest).

The above steps are described further in the next paragraph.

## Example troubleshooting scenario EPG to EPG

This example scenario will show how to troubleshoot a policy drop between two endpoints: 192.168.21.11 in EPG-Web and 192.168.23.11 in EPG-DB.



## Identify the source and destination leaf switches involved in the packet drop

### Visibility & Troubleshooting

The Visibility & Troubleshooting tool will help to visualize the switch where the packet drop occurred for a specific EP-to-EP flow and identify where packets are possibly dropped.

### Configuration of Visibility & Troubleshooting

Visibility & Troubleshooting

**This tool provides:**

1. Location of the specified end points in the fabric and displays the traffic path including any L4-L7 devices. Along the path between these end points, statistics, contracts, faults, events, and audit logs are displayed in scope.
2. Optional triggering of traceroute, and atomic counters for troubleshooting these end points. These debugging steps create and delete corresponding debugging policies as needed.

Session Name:

Session Type:

Description:

**Targets**

Source

Learned At	Tenant	Application	EPG
Pod:1, Leaf:105, Port:eth1/19	Prod1	AppProf	Web

Destination

Learned At	Tenant	Application	EPG
Pod:1, Leaf:105, Port:eth1/19	Prod1	AppProf	DB

Configure a Session Name, Source, and Destination endpoint. Then click 'Submit' or 'Generate Report'.

The tool will automatically find the endpoints in the fabric and provide information about the Tenant, Application Profile and EPG those EP belong to.

In this case, it will discover that the EPs belong to the tenant Prod1, they belong to the same Application Profile 'AppProf' and are assigned to different EPGs: 'Web' and 'DB'.

## Drop identification

The screenshot displays the Cisco APIC interface for drop identification. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Operations' menu is expanded, showing 'Visibility & Troubleshooting', 'Capacity Dashboard', 'EP Tracker', and 'Visualization'. The 'Visibility & Troubleshooting' section is active, showing a search bar and a list of tools: Faults, Drop/Stats, Contracts, Events and Audits, Traceroute, and Atomic Counter. The 'Time Window' is set to 'latest 240 minutes' to 'now'. The 'Session Information' shows a source IP of 192.168.21.11 and a destination IP of 192.168.23.11, with a type of 'Endpoint → Endpoint'. The network topology diagram shows a leaf switch (Leaf fab3-leaf5) connected to a spine switch (Spine fab3-p1-spine1). The leaf switch has two interfaces, eth1/49 and eth1/19, connected to the spine switch's eth1/13. A source endpoint is shown connected to eth1/19. The interface eth1/49 on the leaf switch has a yellow warning icon, and the source endpoint has an information icon. The spine switch interface eth1/13 has a green checkmark icon.

The tool will automatically visualize the topology of the troubleshooting scenario. In this case, the two endpoints happen to be connected to the same leaf switch.

By navigating to the Drop/Stats submenu, the user can view general drops on the leaf or spine in question. Refer to the "Interface Drops" section in the chapter "Intra-Fabric Forwarding" of this book for more information about understanding which drops are relevant.

Many of these drops are expected behavior and can be ignored.

## Drop details

Statistics - fab3-leaf5



Drop Stats

Contract Drops

Traffic Stats

 Show stats with zero values

Time	Affected Object	Stats	Value
2019/10/02 03:49:58 - 2019/10/02 03:54:58	topology/pod-1/node-105/sys/ctx-[vxlan-2654209]/bd-[vxlan-16220082]/vlan-[vlan-701]	ingress drop packets periodic	3
2019/10/02 03:39:48 - 2019/10/02 03:44:58	topology/pod-1/node-105/sys/ctx-[vxlan-2654209]/bd-[vxlan-16121802]/vlan-[vlan-703]	ingress drop packets periodic	3
2019/10/02 03:29:58 - 2019/10/02 03:44:58	topology/pod-1/node-105/sys/ctx-[vxlan-2654209]/bd-[vxlan-16121802]/vlan-[vlan-703]	ingress drop packets periodic	3
2019/10/02 03:29:58 - 2019/10/02 03:44:58	topology/pod-1/node-105/sys/ctx-[vxlan-2654209]/bd-[vxlan-16220082]/vlan-[vlan-701]	ingress drop packets periodic	3
2019/10/02 03:14:58 - 2019/10/02 03:29:58	topology/pod-1/node-105/sys/ctx-[vxlan-2654209]/bd-[vxlan-16121802]/vlan-[vlan-703]	ingress drop packets periodic	3

By drilling down to drop detail using the yellow 'Packets dropped' button on the switch diagram, the user can view details about the dropped flow.

## Contract details

## S Source Endpoint → Destination Endpoint

Filter ID: implicit						BD Allow (Prod1/DB)	
Info	Protocol	L4 Src	L4 Dest	TCP Flags	Action	Nodes	Hits
					permit	node-105	0
Filter ID: implicit						Context Implicit (Prod1/VRF1)	
Info	Protocol	L4 Src	L4 Dest	TCP Flags	Action	Nodes	Hits
					deny,log	node-105	8636

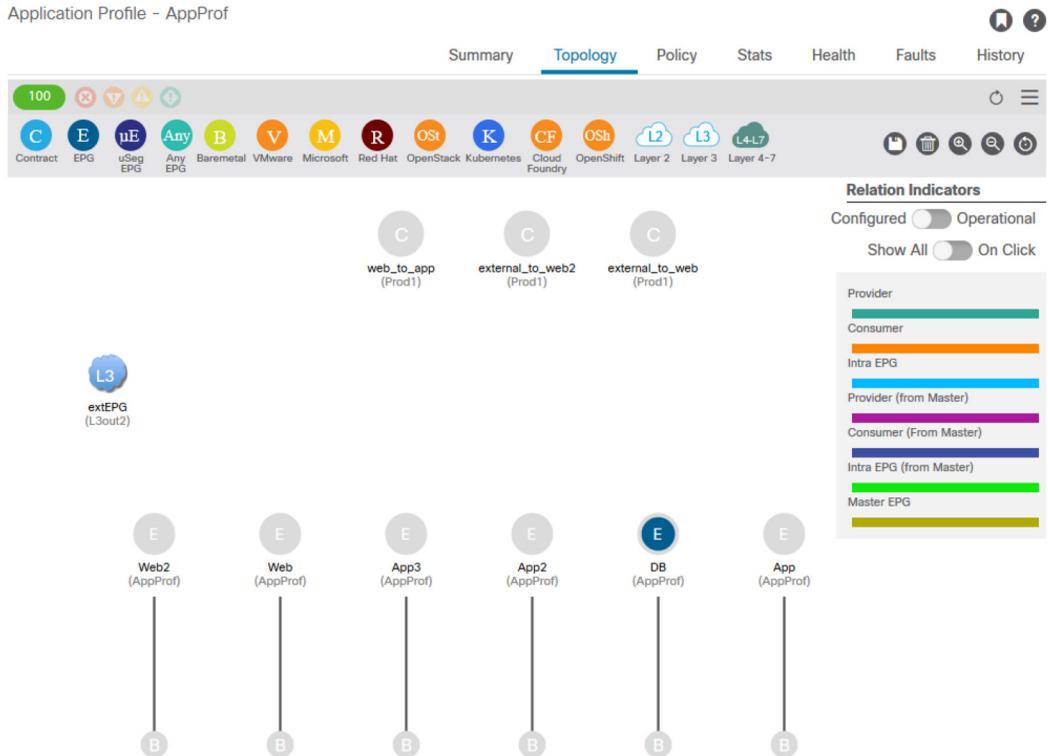
## D Destination Endpoint → Source Endpoint

Filter ID: implicit						BD Allow (Prod1/Web)	
Info	Protocol	L4 Src	L4 Dest	TCP Flags	Action	Nodes	Hits
					permit	node-105	0
Filter ID: implicit						Context Implicit (Prod1/VRF1)	
Info	Protocol	L4 Src	L4 Dest	TCP Flags	Action	Nodes	Hits
					deny,log	node-105	8636

By navigating to the Contracts submenu, the user can identify which contract is causing policy drop off between the EPGs. In the example, it is Implicit to Deny Prod1/VRF1 which shows some hits. This does not necessarily mean the specified flow (192.168.21.11 and 192.168.23.11) is hitting this implicit deny. If the Hits of Context Implicit deny rule is increasing, it implies there is traffic between Prod1/DB and Prod1/Web that do not hit any of contracts, hence are dropped by the Implicit deny.

In the Application Profile Topology view at Tenant > select the Application Profile name on the left > Topology , it is possible to verify which contracts are applied to the DB EPG. In this case, no contract is assigned to the EPG:

### Contract visualization



Now that the source and destination EPGs are known, it is also possible to identify other relevant information such as the following:

- The src/dst **EPG pcTag** of the affected endpoints. The pcTag is the class ID used to identify an EPG with a zoning-rule.
- The src/dst **VRF VNID**, also referred to as **scope**, of the affected endpoints.

Class ID and scope can be easily retrieved from the APIC GUI by opening the Tenant > select the Tenant name on the left > Operational > Resource IDs > EPGs

### Tenant resource ID to find EPG pcTag and scope

Tenant - Prod1

Summary Dashboard Policy **Operational** Stats Health Faults History

Flows Packets **Resource IDs**

Bridge Domains VRFs **EPGs** L3Outs External Networks (Bridged)

Application Profile Name	AP Alias	EPG Name	Class ID	Scope
AppProf		App	32774	2654209
AppProf		App2	32775	2654209
AppProf		App3	49160	2654209
AppProf		DB	49159	2654209
AppProf		Web	32778	2654209
AppProf		Web2	16388	2097160
Services		NTP	16410	2818048

In this case the Class ID and Scopes are:

- Web EPG pcTag 32778
- Web EPG scope 2654209
- DB EPG pcTag 49159
- DB EPG scope 2654209

## Verify the policy applied to the traffic flow being troubleshoot

### iBash

An interesting tool to verify the packet dropped on an ACI leaf is the iBash command line: 'show logging ip access-list internal packet-log deny':

```
leaf5# show logging ip access-list internal packet-log deny | grep 192.168.21.11
[2019-10-01T14:25:44.746528000+09:00]: CName: Prod1:VRF1(VXLAN: 2654209), VlanType: FD_VLAN, Vlan-Id: 114, SMac: 0xf6f26c4ec8d0,
DMac:0x0022bdf819ff, SIP: 192.168.21.11, DIP: 192.168.23.11, SPort: 0, DPort: 0, Src Intf: Ethernet1/19, Proto: 1, PktLen: 126
[2019-10-01T14:25:44.288653000+09:00]: CName: Prod1:VRF1(VXLAN: 2654209), VlanType: FD_VLAN, Vlan-Id: 116, SMac: 0x3e2593f0eded,
DMac:0x0022bdf819ff, SIP: 192.168.23.11, DIP: 192.168.21.11, SPort: 0, DPort: 0, Src Intf: Ethernet1/19, Proto: 1, PktLen: 126
```

As per the previous output, it can be seen that on the leaf switch, numerous ICMP packets sourced by EP 192.168.23.11 towards 192.168.21.11 have been dropped.

The contract\_parser tool will help to verify the actual policies applied to the VRF where the Endpoints are associated with:

```
leaf5# contract_parser.py --vrf Prod1:VRF1
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-14] dst-epg [dst-14] [flags][contract:{str}] [hit=count]

[7:5159] [vrf:Prod1:VRF1] permit ip tcp tn-Prod1/ap-App1/epg-App(32771) eq 5000 tn-Prod1/ap-App1/epg-Web(32772) [contract:uni/tn-Prod1/brc-web-to_app] [hit=0]
[7:5156] [vrf:Prod1:VRF1] permit ip tcp tn-Prod1/ap-App1/epg-Web(32772) tn-Prod1/ap-App1/epg-App(32771) eq 5000 [contract:uni/tn-Prod1/brc-web-to_app] [hit=0]
[16:5152] [vrf:Prod1:VRF1] permit any epg:any tn-Prod1/bd-Web(49154) [contract:implicit] [hit=0]
[16:5154] [vrf:Prod1:VRF1] permit arp epg:any epg:any [contract:implicit] [hit=0]
[21:5155] [vrf:Prod1:VRF1] deny,log any epg:any epg:any [contract:implicit] [hit=38,+10]
[22:5153] [vrf:Prod1:VRF1] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]
```

This can also be verified through the zoning rule programmed in the leaf the policies enforced by the switch.

```
leaf5# show zoning-rule scope 2654209
```

Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name	Action	Priority
5155	0	0	implicit	uni-dir	enabled	2654209		deny_log	any_any_any(21)
5159	32771	32772	411	uni-dir-ignore	enabled	2654209	web_to_app	permit	fully_qual(7)
5156	32772	32771	410	bi-dir	enabled	2654209	web_to_app	permit	fully_qual(7)

As already seen by the Visibility & Troubleshooting tool, the `contract_parser` tool, and the zoning rules, output confirms there is no contract between the source and destination EPGs in troubleshooting. It is easy to assume that the packets dropped are matching the implicit deny rule 5155.

## ELAM Capture

ELAM capture provides an ASIC level report used to check forwarding details which indicates, in the case of a dropped packet, the drop reason. When the reason of a drop is a policy drop, as in this scenario, the output of the ELAM capture will look like the following.

Please note that details of setting up an ELAM capture will not be discussed in this chapter, please see the chapter "Intra-Fabric Forwarding".

```
leaf5# vsh_lc
module-1# debug platform internal tah elam ASIC 0
module-1(DBG-elam)# trigger init in-select 6 out-select 0
module-1(DBG-elam)# trigger reset
module-1(DBG-elam-insel6)# set outer ipv4 src_ip 192.168.21.11 dst_ip 192.168.23.11
module-1(DBG-elam-insel6)# start
module-1(DBG-elam-insel6)# status

ELAM STATUS
=====
Asic 0 Slice 0 Status Triggered
Asic 0 Slice 1 Status Armed

module-1(DBG-elam-insel6)# ereport | grep reason
```

```

RW drop reason          : SECURITY_GROUP_DENY
LU drop reason          : SECURITY_GROUP_DENY
pkt.lu_drop_reason: 0x2D

```

The ELAM report above shows clearly that the packet was dropped due to a policy drop: 'SECURITY\_GROUP\_DENY'

### ELAM Assistant:

The very same result of the ELAM capture can be shown through the ELAM Assistant App on the APIC GUI.

## Configuration

Configuration interface for ELAM (Embedded Logic Analyzer Module) capture.

Header: Capture a packet with ELAM (Embedded Logic Analyzer Module)

ELAM PARAMETERS

Name your capture:

Parameters	Status	Node	Direction	Source I/F
			VxLAN (outer) header	
	Report Ready	node-105	from frontport	eth1/19
src ip		<input type="text" value="192.168.21.11"/>		
dst ip		<input type="text" value="192.168.23.11"/>		

Buttons:

Typically, the user will configure both source and destination details for the flow of interest. In this example, src IP is used to capture traffic towards endpoint in destination EPG that does not have a contract relationship to the source EPG.

## Elam Assistant Express report

**ELAM Report Parse Result ( report name: node-105\_slot1\_asic0\_elam\_report.txt )**

**Express**
Detail
Raw

There are three levels of output that can be viewed with ELAM Assistant. These are Express, Detail, and Raw.

## Elam Assistant Express report (cont.)

**Packet Forwarding Information**

Forward Result	
Destination Type	To a local port
Destination Logical Port	Eth1/19
Destination Physical Port	packet dropped
Sent to SUP/CPU instead	yes
SUP Redirect Reason (SUP code)	ISTACK_SUP_CODE_ACL_LOG
Contract	
Destination EPG pcTag (dclass)	16387 (Prod1:App1:DB)
Source EPG pcTag (sclass)	10935 (Prod1:App1:Web)
Contract was applied	0 (Contract was not applied on this node)
Drop	
Drop Code	SECURITY_GROUP_DENY

Under the Express Result, the Drop Code reason SECURITY\_GROUP\_DENY indicates that the drop was a result of a contract hit.

# Preferred group

## 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 deny rules between the excluded EPG and any EPG apply.

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, configure a combination of a contract preferred group and contracts with filters to more precisely control inter-EPG communication.

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.

## Contract Preferred Group programming

Essentially, Contract Preferred Groups are an inverse of regular contracts. For regular contracts, explicit permit zoning-rules are programmed with an implicit deny zoning-rule with the VRF Scope. For Preferred Groups, an implicit PERMIT zoning-rule is programmed with the highest numeric priority value and specific DENY zoning-rules are programmed to disallow traffic from EPGs which are not Preferred Group members. As a result, the deny rules are evaluated first and if the flow isn't matched by these rules, then the flow is implicitly permitted.

There's always a pair of explicit deny zoning-rules for every EPG outside of the preferred group:

- One from the non-Preferred Group member to any pcTag (value 0).
- Another from any pcTag (value 0) to the non-Preferred Group member.

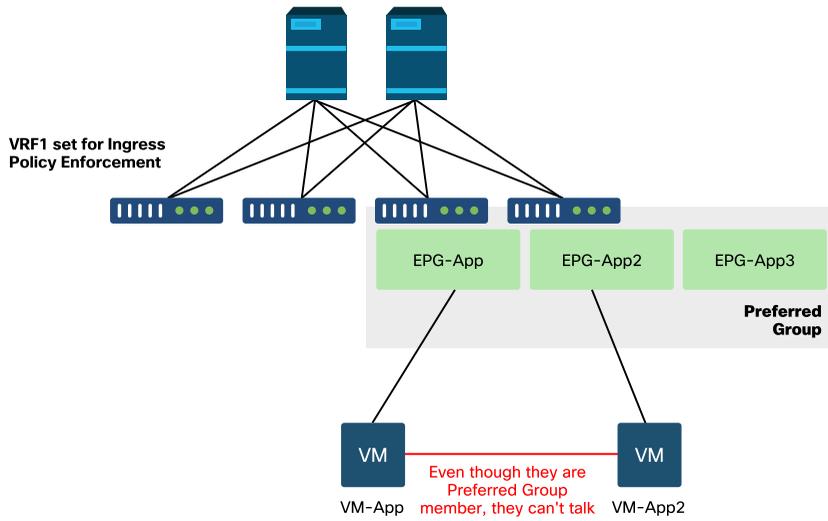
## Scenario

The figure below shows a logical topology in which EPGs App, App2 and App3 are all configured as Preferred Group Members.

VM-App is part of EPG-App and VM-App2 is part of EPG-App2. Both App and App2 EPG should be part of the preferred and hence communicate freely.

VM-App initiates a traffic flow on TCP port 6000 to VM-App2. Both EPG-App and EPG-App2 are Preferred Group Members as part of VRF1. VM-App2 never receives any packets on TCP port 6000.

### Troubleshooting scenario



## Workflow

### 1. Look up the pcTag of EPG APP and its VRF VNID/Scope

#### EPG and VRF pcTags

The screenshot shows the APIC interface for the 'Prod1' tenant. The 'Operational' tab is selected, and the 'EPGs' section is expanded. A table lists application profiles with their names, aliases, EPG names, class IDs, and scopes. Red boxes highlight the 'App' and 'App2' entries, and their corresponding class IDs and scopes.

Application Profile Name	AP Alias	EPG Name	Class ID	Scope
AppProf		App	32774	2654209
AppProf		App2	32775	2654209
AppProf		App3	49160	2654209
AppProf		DB	49159	2654209
AppProf		Web	16390	2654209
AppProf		Web2	16388	2097160

### 2. Verify contract programming using contract\_parser.py on the ingress leaf

Use contract\_parser.py and/or the 'show zoning-rule' command and specify the VRF

```
fab3-leaf8# show zoning-rule scope 2654209
```

Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name	Action	Priority
4165	0	0	implicit	uni-dir	enabled	2654209		permit	grp-any-any-permit(20)
4160	0	0	implarp	uni-dir	enabled	2654209		permit	any-any-filter(17)
4164	0	15	implicit	uni-dir	enabled	2654209		deny,log	grp-any_dest-any_deny(19)
4176	0	16386	implicit	uni-dir	enabled	2654209		permit	any_dest-any(16)
4130	32770	0	implicit	uni-dir	enabled	2654209		deny,log	grp_src-any-any_deny(18)
4175	49159	0	implicit	uni-dir	enabled	2654209		deny,log	grp_src-any-any_deny(18)
4129	0	49159	implicit	uni-dir	enabled	2654209		deny,log	grp-any_dest-any_deny(19)
4177	32778	0	implicit	uni-dir	enabled	2654209		deny,log	grp_src-any-any_deny(18)
4128	0	32778	implicit	uni-dir	enabled	2654209		deny,log	grp-any_dest-any_deny(19)
4178	32775	0	implicit	uni-dir	enabled	2654209		deny,log	grp_src-any-any_deny(18)

```

| 4179 | 0 | 32775 | implicit | uni-dir | enabled | 2654289 | | deny,log | grp_any_dest_any_deny(19) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
fab3-leaf8# contract_parser.py --vrf Prod1:VRF1
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-14] dst-epg [dst-14] [flags][contract:{str}] [hit=count]
[16:4176] [vrf:Prod1:VRF1] permit any epg:any tn-Prod1/bd-App(16386) [contract:implicit] [hit=0]
[16:4160] [vrf:Prod1:VRF1] permit arp epg:any epg:any [contract:implicit] [hit=0]
[18:4130] [vrf:Prod1:VRF1] deny,log any tn-Prod1/vrf-VRF1(32770) epg:any [contract:implicit] [hit=?]
[18:4178] [vrf:Prod1:VRF1] deny,log any epg:32775 epg:any [contract:implicit] [hit=?]
[18:4177] [vrf:Prod1:VRF1] deny,log any epg:32778 epg:any [contract:implicit] [hit=?]
[18:4175] [vrf:Prod1:VRF1] deny,log any epg:49159 epg:any [contract:implicit] [hit=?]
[19:4164] [vrf:Prod1:VRF1] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]
[19:4179] [vrf:Prod1:VRF1] deny,log any epg:any epg:32775 [contract:implicit] [hit=?]
[19:4128] [vrf:Prod1:VRF1] deny,log any epg:any epg:32778 [contract:implicit] [hit=?]
[19:4129] [vrf:Prod1:VRF1] deny,log any epg:any epg:49159 [contract:implicit] [hit=?]
[20:4165] [vrf:Prod1:VRF1] permit any epg:any epg:any [contract:implicit] [hit=65]

```

Examining the above output, the implicit permit entry — ruleId 4165 — with the highest priority of 20, is observed. This implicit permit rule will cause all traffic flows to be allowed unless there's an explicit deny rule with a lower priority disallowing the traffic flow.

In addition, there are two explicit deny rules observed for pcTag 32775 which is the pcTag of EPG App2. These two explicit deny zoning-rules disallow traffic from any EPG to EPG App2, and vice versa. Those rules have priority 18 and 19, so they will take precedence on the default permit rule.

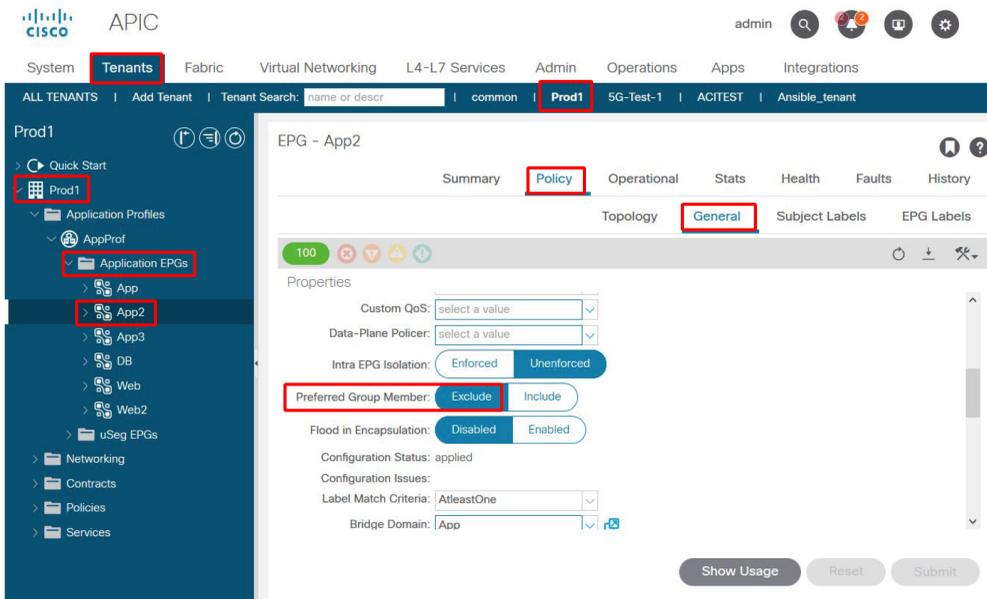
The conclusion is that EPG App2 is not a Preferred Group Member as the explicit deny rules are observed.

### 3. Verify EPG preferred Group Member Configuration

Navigate the APIC GUI and check EPG App2 and EPG App Preferred Group Member Configuration,

In the following figure, see EPG App2 is not configured as a Preferred Group Member.

EPG App2 – Preferred Group Member setting excluded



## EPG App – Preferred Group Member setting included

The screenshot displays the Cisco APIC (Application Policy Infrastructure Controller) interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Tenants' tab is active, and the 'Prod1' tenant is selected. The left sidebar shows a tree view of the tenant's configuration, with 'Application EPGs' expanded to show 'App' selected. The main content area is titled 'EPG - App' and has tabs for 'Summary', 'Policy', 'Operational', 'Stats', 'Health', 'Faults', and 'History'. The 'Policy' tab is active, and the 'General' sub-tab is selected. The 'Properties' section shows various configuration options: 'Custom QoS' (select a value), 'Data-Plane Policer' (select a value), 'Intra EPG Isolation' (Enforced/Unenforced), 'Preferred Group Member' (Exclude/Include), 'Flood in Encapsulation' (Disabled/Enabled), 'Configuration Status' (applied), 'Configuration Issues', 'Label Match Criteria' (AtleastOne), and 'Bridge Domain' (App). The 'Preferred Group Member' setting is highlighted with a red box, and the 'Include' radio button is selected. At the bottom right, there are buttons for 'Show Usage', 'Reset', and 'Submit'.

#### 4. Set EPG App2 to be a Preferred Group Member

Changing the configuration of App2 EPG enables the preferred group to communicate freely as part of the preferred group.

#### EPG App2 – Preferred Group Member setting included

The screenshot shows the APIC interface for configuring EPG App2. The 'Tenants' tab is selected, and the 'Prod1' tenant is active. The 'EPG - App2' configuration page is open, with the 'Policy' and 'General' tabs selected. The 'Preferred Group Member' setting is set to 'Include'.

#### 5. Re-verify contract programming using contract\_parser.py on the leaf where the src EP resides

Use contract\_parser.py again and specify the VRF name to verify whether the explicit deny rules for EPG App2 are now gone.

```
fab3-leaf8# contract_parser.py --vrf Prod1:VRF1
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-14] dst-epg [dst-14] [flags][contract:{str}] [hit=count]
[16:4176] [vrf:Prod1:VRF1] permit any epg:any tn-Prod1/bd-App(16386) [contract:implicit] [hit=0]
[16:4160] [vrf:Prod1:VRF1] permit arp epg:any epg:any [contract:implicit] [hit=0]
[18:4175] [vrf:Prod1:VRF1] deny,log any epg:16390 epg:any [contract:implicit] [hit=0]
[18:4167] [vrf:Prod1:VRF1] deny,log any epg:23 epg:any [contract:implicit] [hit=0]
[18:4156] [vrf:Prod1:VRF1] deny,log any tn-Prod1/vrf-VRF1(32770) epg:any [contract:implicit] [hit=0]
[18:4168] [vrf:Prod1:VRF1] deny,log any epg:49159 epg:any [contract:implicit] [hit=0]
[19:4164] [vrf:Prod1:VRF1] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]
[19:4169] [vrf:Prod1:VRF1] deny,log any epg:any epg:16390 [contract:implicit] [hit=0]
[19:4159] [vrf:Prod1:VRF1] deny,log any epg:any epg:23 [contract:implicit] [hit=0]
```

```
[19:4174] [vrf:Prod1:VRF1] deny,log any epg:any epg:49159 [contract:implicit] [hit=0]  
[20:4165] [vrf:Prod1:VRF1] permit any epg:any epg:any [contract:implicit] [hit=65]
```

The explicit deny rules for EPG App2 and its pcTag 32775 are no longer observed in the above output. This means that traffic between EPs in EPG App and EPG App2 will now match the implicit permit rule – ruleId 4165 – with the highest priority of 20.

## vzAny to EPG

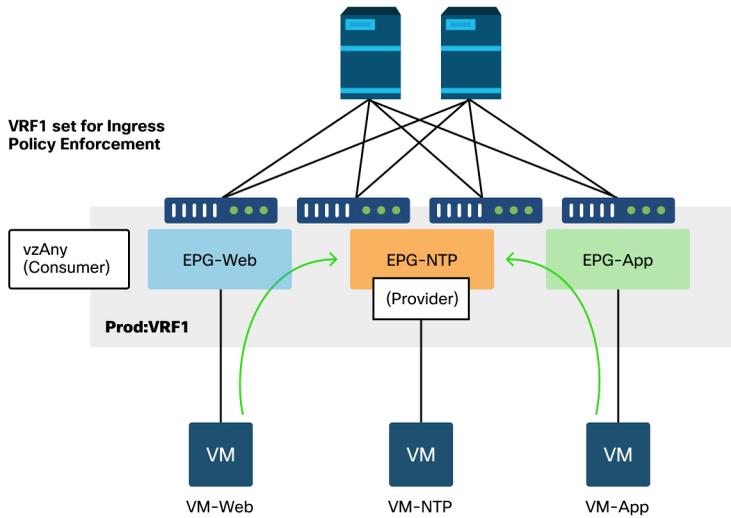
### About vzAny

When configuring contracts between one or multiple EPGs, contracts can either be configured as a consumed or provided relation. When the number of EPGs grows, so can the amount of contract relations between them. Some common use cases require all EPGs to exchange traffic flows with another specific EPG. Such a use case could be an EPG containing EPs providing services that need to be consumed by all other EPGs inside the same VRF (NTP or DNS for example). vzAny allows for lower operational overhead in configuring contract relations between all EPGs and specific EPGs providing services to be consumed by all other EPGs. In addition, vzAny allows for a much more efficient Security Policy CAM usage on leaf switches as only 2 zoning-rules are added for each vzAny contract relation.

### Scenarios

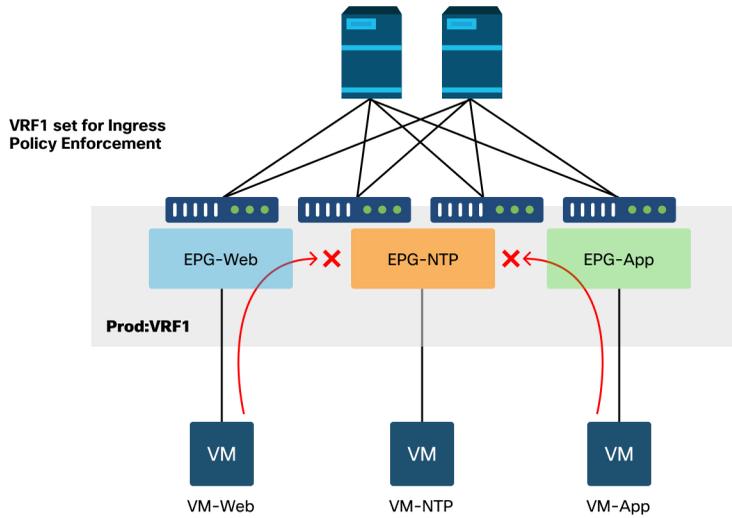
The figure below describes such a use case whereby VM-Web and VM-App in EPGs Web and App respectively need to consume NTP services from VM-NTP in EPG-NTP. Instead of configuring a provided contract on EPG NTP, and subsequently having that same contract as a consumed contract on EPGs Web and App, vzAny allows each EPG in VRF Prod:VRF1 to consume NTP services from EPG NTP.

vzAny – Any EPG in VRF Prod:VRF1 can consume NTP services from EPG NTP



Consider a scenario where drops are observed between EPGs that consume the NTP services when there is no contract between them.

Traffic drops if there is no contract



## Workflow

### 1. Look up the pcTag of EPG NTP and its VRF VNID/Scope

'Tenant > Operational > Resource IDs > EPGs' allows finding the pcTag and scope

#### EPG NTP pcTag and its VRF VNID/Scope

Tenant - Prod1

Summary Dashboard Policy **Operational** Stats Health Faults History

Flows Packets **Resource IDs**

Bridge Domains VRFs **EPGs** L3Outs External Networks (Bridged)

Application Profile Name	AP Alias	EPG Name	Class ID	Scope
AppProf		App	32774	2654209
AppProf		App2	32775	2654209
AppProf		App3	49160	2654209
AppProf		DB	49159	2654209
AppProf		Web	32778	2654209
AppProf		Web2	16388	2097160
Services		NTP	16410	2818048

Page 1 Of 1 Objects Per Page: 100 Displaying Objects 1 - 7 Of 7

**2. Verify if a contract is configured as a vzAny consumed contract as part of the VRF**  
Navigate to the VRF and check if there's a consumed contract configured as vzAny under the 'EPG Collection for VRF'.

Contract configured as a consumed vzAny contract on the VRF

The screenshot displays the Cisco APIC interface for configuring a VRF. The left-hand navigation pane shows the hierarchy: Prod1 > Networking > VRFs > VRF1 > EPG Collection for VRF. The main content area shows the configuration for 'vzAny' under the 'Prod1' tenant. The 'General' tab is selected, and the 'Consumed Contracts' table is visible. The table contains one entry: 'any\_to\_ntp' with Tenant 'Prod1', Type 'Contract', QoS Class 'Unspecified', and State 'formed'. The 'Policy' and 'General' tabs are also visible at the top of the configuration area.

Consumed Contracts:					
Name	Tenant	Type	QoS Class	State	
any_to_ntp	Prod1	Contract	Unspecified	formed	

### 3. Verify if the same contract is applied as a provided contract on EPG NTP

In order to establish a contract relation, the same contract needs to be applied as a provided contract on EPG NTP which is providing NTP services to the other EPGs in its VRF.

Contract configured as a provided contract on EPG NTP

The screenshot shows the Cisco APIC interface. The left sidebar displays the navigation tree for 'Prod1', with 'Contracts' highlighted. The main content area shows the 'Contracts' configuration page. A table lists the configured contracts, with the following data:

Tenant Name	Tena Alias	Contract Name	Contract Type	Provided / Consumed	QoS Class	State	Label	Subject Label
Prod1		any_to_ntp	Contra...	Provid...	Unspecified	formed		

### 4. Zoning-rule verification on ingress leaf using contract\_parser.py or 'show zoning-rule'

The ingress leaf should have 2 zoning-rules to allow bi-directional traffic flows (if the contract subject is set to allow both directions) between any EPG and EPG NTP. 'Any EPG' is denoted as pcTag 0 in zoning-rule programming.

Using contract\_parser.py or the 'show zoning-rule' commands on the ingress leaf whilst specifying the VRF allows to ensure the zoning-rule are programmed.

**Zoning-rules allowing traffic to/from EPG NTP from other EPGs in the VRF present**

Using `contract_parser.py` and 'show zoning-rule' to check the presence of the vzAny based zoning-rules.

Here two types of rules are evident:

- 1 Rule 4156 and Rule 4168 which permit Any to NTP and vice-versa. They have priority 13 and 14:
  - Zoning-rule allowing traffic flows from any EPG (pcTag 0) to EPG NTP (pcTag 49161).
  - Zoning-rule allowing traffic flows from EPG NTP (pcTag 46161) to any other EPG (pcTag 0).
- 2 Rule 4165 which is the any to any deny rule (default) with priority 21.

Given that lowest priority has precedence, all EPGs of the VRF will have access NTP EPG.

```
fab3-leaf8# contract_parser.py --vrf Prod1:VRF
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-14] dst-epg [dst-14] [flags][contract:{str}] [hit=count]

[13:4156] [vrf:Prod1:VRF1] permit ip tcp tn-Prod1/ap-Services/epg-NTP(49161) eq 123 epg:any [contract:uni/tn-Prod1/brc-any_to_ntp] [hit=0]
[14:4168] [vrf:Prod1:VRF1] permit ip tcp epg:any tn-Prod1/ap-Services/epg-NTP(49161) eq 123 [contract:uni/tn-Prod1/brc-any_to_ntp] [hit=0]
[16:4176] [vrf:Prod1:VRF1] permit any epg:any tn-Prod1/bd-App(16386) [contract:implicit] [hit=0]
[16:4174] [vrf:Prod1:VRF1] permit any epg:any tn-Prod1/bd-Services(32776) [contract:implicit] [hit=0]
[16:4160] [vrf:Prod1:VRF1] permit arp epg:any epg:any [contract:implicit] [hit=0]
[21:4165] [vrf:Prod1:VRF1] deny,log any epg:any epg:any [contract:implicit] [hit=65]
[22:4164] [vrf:Prod1:VRF1] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]
```

```
fab3-leaf8# show zoning-rule scope 2654209
```

Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name	Action	Priority
<b>4165</b>	<b>0</b>	<b>0</b>	<b>implicit</b>	<b>uni-dir</b>	<b>enabled</b>	<b>2654209</b>		<b>deny,log</b>	<b>any_any_any(21)</b>
4160	0	0	implarp	uni-dir	enabled	2654209		permit	any_any_filter(17)
4164	0	15	implicit	uni-dir	enabled	2654209		deny,log	any_vrf_any_deny(22)
4176	0	16386	implicit	uni-dir	enabled	2654209		permit	any_dest_any(16)
4174	0	32776	implicit	uni-dir	enabled	2654209		permit	any_dest_any(16)
<b>4168</b>	<b>0</b>	<b>49161</b>	<b>424</b>	<b>uni-dir</b>	<b>enabled</b>	<b>2654209</b>	<b>any_to_ntp</b>	<b>permit</b>	<b>any_dest_filter(14)</b>
<b>4156</b>	<b>49161</b>	<b>0</b>	<b>425</b>	<b>uni-dir</b>	<b>enabled</b>	<b>2654209</b>	<b>any_to_ntp</b>	<b>permit</b>	<b>src_any_filter(13)</b>

# Shared L3Out to EPG

## Description

Shared Layer 3 Out is a configuration which allows having an L3Out in one VRF providing some services (external access) and one or more other VRFs consume this L3Out. More detail on shared L3Out can be found in the "External routing" chapter.

When doing shared L3Out it is recommended to have the provider of the contract being the shared L3Out and the EPG being the consumer of the contract. This scenario will be illustrated in this section.

It is not recommended to do the opposite, which is L3Out consuming a service provided by an EPG. This configuration leads to less scalability and that scenario will not be illustrated here.

## Scenario: provided contract on L3Out EPG and consumed contract on EPG

### Motivation

The recommended configuration when requiring traffic flows between an L3Out and an EPG in different VRFs is to configure the external EPG (l3extInstP) as the provider and the EPG as the consumer. The behavior is such that zoning-rules are only installed on consumer VRF. The principles of consuming and providing denote where traffic flows are initiated. With default ingress policy enforcement, this means policy enforcement will be applied on the consumer side and more specifically on the ingress leaf (non-border leaf). For the ingress leaf to enforce policy it requires the pcTag of the destination. In this scenario the destination is the external EPG pcTag. The ingress leaf thus performs policy enforcement and forwards the packets to the border leaf. The border leaf receives the packet on its fabric link which performs a route lookup (LPM) and forwards the packet onto the adjacency for the destination prefix.

The border leaf however does NOT perform any policy enforcement when sending traffic onto the destination EP nor does it do so on the return traffic flow back to the source EP.

As a result, only the Policy CAM of the ingress non-BL leaf has entries installed (in the consumer VRF) and the BL's Policy CAM is not affected.

### 1. Verify EPG pcTag and VRF VNID/Scope for the consumer EPG

With shared L3Out, the zoning-rules are only installed in the consumer VRF. The provider must have a global pcTag (below 16k) which allows this pcTag to be used in all consumer VRFs. In our scenario, the provider is the external EPG and will have a global pcTag. The consumer EPG will have a local pcTag as usual.

#### pcTag of consumer EPG

Tenant - Prod1

Summary Dashboard Policy **Operational** Stats Health Faults History

Flows Packets **Resource IDs**

Bridge Domains VRFs **EPGs** L3Outs External Networks (Bridged)

Application Profile Name	AP Alias	EPG Name	Class ID	Scope
AppProf		App	32774	2654209
AppProf		App2	32775	2654209
AppProf		App3	49160	2654209
AppProf		DB	49159	2654209
AppProf		Web	32778	2654209
AppProf		Web2	16388	2097160
Services		NTP	16410	2818048

Page 1 Of 1 Objects Per Page: 100 Displaying Objects 1 - 7 Of 7

## 2. Verify the pcTag and VRF VNID/Scope for the provider L3Out EPG

As noted in Step 1, the provider L3Out EPG has a global range pcTag as prefixes from L3Out which are leaked into the consumer VRF. As a result, the L3Out EPG pcTag is required to not overlap with pcTags in the consumer VRF, and so it is within the global pcTag range.

### pcTag of provider external EPG

Tenant - Prod1

Summary Dashboard Policy **Operational** Stats Health Faults History

Flows Packets Resource IDs

Bridge Domains VRFs EPGs **L3Outs** External Networks (Bridged)

EPG Name	EPG Alias	Class ID	Scope
extEpg		25	2719752

Page 1 Of 1 Objects Per Page: 100 Displaying Objects 1 - 1 Of 1

### 3. Verify the consumer EPG has either an imported tenant scoped contract or global contract configured

The consumer EPG NTP with subnet defined under the EPG/BD is consuming the 'tenant' or 'global' scoped contract

#### Contract consumed by EPG

The screenshot shows the Cisco APIC interface for tenant 'Prod1'. The left sidebar shows the navigation tree with 'Contracts' highlighted under 'Application EPGs'. The main content area displays a table of contracts consumed by the EPG.

Tenar Name	Tena Alias	Contract Name	Contract Type	Provided / Consumed	QoS Class	State	Label	Sub Lab
Prod1	external_to_ntp	Contract	Contract	Consumed	Unspecified	form...		

### 4. Verify whether the BD of the consumer EPG has a subnet configured with its scope set to 'Shared between VRFs'

The subnet of the EPG is configured under the bridge domain but must have the 'shared between VRF' flag (to allow routed leaking) and the 'advertised externally' flag (to allow to advertise to L3Out)

### 5. Verify the provider L3Out EPG has either an imported tenant scoped contract or global contract configured

The L3Out EPG should either have a tenant scoped contract or global contract configured as a provided contract.

## Contract on provider L3Out

The screenshot displays the Cisco APIC interface for configuring a contract on a provider L3Out EPG. The navigation menu on the left shows the path: Prod1 > L3Outs > L3Out1 > External EPGs > extEpg. The top navigation bar shows 'Tenants' and 'Prod1' selected. The main content area is titled 'External EPG Instance Profile - extEpg' and has tabs for 'Policy', 'Operational', 'Stats', 'Health', 'Faults', and 'History'. The 'Contracts' tab is selected, showing a sub-tab for 'Provided Contracts'. Below this, a table lists the provided contracts:

Name	Tenant	Type	QoS Class	Match Type	State
external_to_ntp	Prod1	Contract	Unspecified	AtleastOne	formed

### 6. Verify if the provider L3Out EPG has a subnet configured with the necessary scopes checked

The provider L3Out EPG should have the to-be-leaked prefix configured with the following scopes:

- External subnets for the external EPG.
- Shared route control subnet.
- Shared security import subnet.

For more detail on subnet flag in L3Out EPG refer to the "External forwarding" chapter.

### External EPG subnet settings

The screenshot shows the APIC interface for configuring an External EPG Instance Profile named 'extEpg'. The 'Policy' and 'General' tabs are selected. The 'Subnets' table is expanded to show the following entry:

IP Address	Scope	Name	Aggregate	Route Control Profile	Route Summary Policy
172.16.10.0/24	External Subnets for the Shared Route Control S...	Shared Security Import ...			

### External EPG subnet settings expanded

The screenshot shows the expanded configuration for the subnet '172.16.10.0/24'. The 'Policy' tab is active, and the following settings are visible:

- IP Address: 172.16.10.0/24 (address/mask)
- Scope:
  - Export Route Control Subnet
  - Import Route Control Subnet
  - External Subnets for the External EPG
  - Shared Route Control Subnet
  - Shared Security Import Subnet
- Aggregate:
  - Aggregate Export
  - Aggregate Import
  - Aggregate Shared Routes

### 7. Verify the pcTag of L3Out EPG subnet on the non-BL for the consumer VRF

When traffic destined to the external EPG subnet ingresses the non-BL, a lookup is performed against the destination prefix to determine the pcTag. This can be checked using the following command on the non-BL.

Note this output is taken in the scope of the VNI 2818048 which is the consumer VRF VNID. By looking at the table the consumer can find the pcTag of the destination, even though it is not in the same VRF.

```
fab3-leaf8# vsh -c 'show system internal policy-mgr prefix' | egrep 'Vrf-Vni|==|common:default'
```

Vrf-Vni	VRF-Id	Table-Id	Table-State	VRF-Name	Addr	Class	Shared	Remote	Complete
2818048	19	0x13	Up	common:default	0.0.0.0/0	15	False	False	False
2818048	19	0x80000013	Up	common:default	::/0	15	False	False	False
<b>2818048</b>	<b>19</b>	<b>0x13</b>	<b>Up</b>	<b>common:default</b>	<b>172.16.10.0/24</b>	<b>25</b>	<b>True</b>	<b>True</b>	<b>False</b>

The above output shows the combination of the L3Out EPG subnet and its global pcTag 25.

### 8. Verify the programmed zoning-rules on the non-BL for the consumer VRF

Use either 'contract\_parser.py' or the 'show zoning-rule' command and specify the VRF.

Below command outputs display two zoning-rules are installed to allow traffic from the consumer EPG local pcTag 16410 to the L3Out EPG global pcTag 25. This is in the scope 2818048, which is the scope of the consumer VRF.

```
fab3-leaf8# show zoning-rule scope 2818048
```

Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name	Action	Priority
4174	0	0	implarp	uni-dir	enabled	2818048		permit	any_any_filter(17)
4168	0	15	implicit	uni-dir	enabled	2818048		deny_log	any_vrf_any_deny(22)
4167	0	32789	implicit	uni-dir	enabled	2818048		permit	any_dest_any(16)
4159	0	0	implicit	uni-dir	enabled	2818048		deny_log	any_any_any(21)
4169	25	0	implicit	uni-dir	enabled	2818048		deny_log	shsrc_any_any_deny(12)
<b>4156</b>	<b>25</b>	<b>16410</b>	<b>425</b>	<b>uni-dir-ignore</b>	<b>enabled</b>	<b>2818048</b>	<b>external_to_ntp</b>	<b>permit</b>	<b>fully_qual(7)</b>
<b>4131</b>	<b>16410</b>	<b>25</b>	<b>424</b>	<b>bi-dir</b>	<b>enabled</b>	<b>2818048</b>	<b>external_to_ntp</b>	<b>permit</b>	<b>fully_qual(7)</b>

```
fab3-leaf8# contract_parser.py --vrf common:default
```

Key:

```
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-14] dst-epg [dst-14] [flags][contract:{str}] [hit=count]
```

```
[7:4131] [vrf:common:default] permit ip tcp tn-Prod1/ap-Services/epg-NTP(16410) tn-Prod1/13out-L3Out1/instP-extEpg(25) eq 123
[contract:uni/tn-Prod1/brc-external_to_ntp] [hit=0]
[7:4156] [vrf:common:default] permit ip tcp tn-Prod1/13out-L3Out1/instP-extEpg(25) eq 123 tn-Prod1/ap-Services/epg-NTP(16410)
[contract:uni/tn-Prod1/brc-external_to_ntp] [hit=0]
[12:4169] [vrf:common:default] deny,log any tn-Prod1/13out-L3Out1/instP-extEpg(25) epg:any [contract:implicit] [hit=0]
[16:4167] [vrf:common:default] permit any epg:any tn-Prod1/bd-Services(32789) [contract:implicit] [hit=0]
[16:4174] [vrf:common:default] permit arp epg:any epg:any [contract:implicit] [hit=0]
[21:4159] [vrf:common:default] deny,log any epg:any epg:any [contract:implicit] [hit=0]
[22:4168] [vrf:common:default] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]
```

## 9. Verify the programmed zoning-rules on the BL for the provider VRF

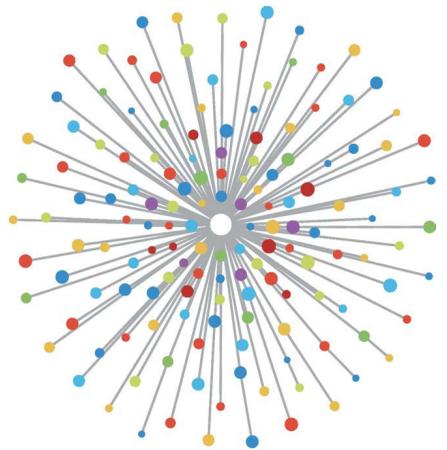
Use either 'contract\_parser.py' or the 'show zoning-rule' command and specify the VRF. The following command outputs show that there are **NO** specific zoning-rules in the provider VRF as outlined multiple times before.

It is in the scope 2719752 which is the scope of provider VRF.

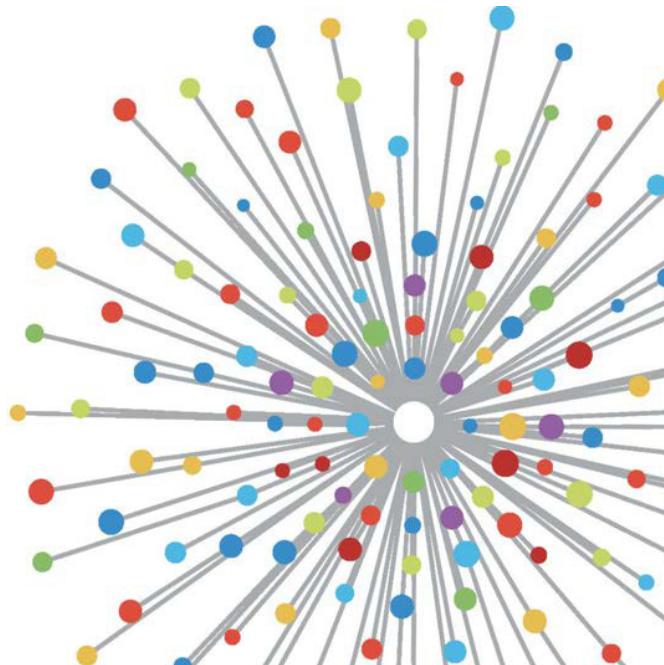
```
border-leaf# show zoning-rule scope 2719752
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir | operSt | Scope | Name | Action | Priority |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 4134 | 10937 | 24 | default | uni-dir-ignore | enabled | 2719752 | vrf1_to_vrf2 | permit | src_dst_any(9) |
| 4135 | 24 | 10937 | default | bi-dir | enabled | 2719752 | vrf1_to_vrf2 | permit | src_dst_any(9) |
| 4131 | 0 | 0 | implicit | uni-dir | enabled | 2719752 | | deny,log | any_any_any(21) |
| 4130 | 0 | 0 | implarp | uni-dir | enabled | 2719752 | | permit | any_any_filter(17) |
| 4132 | 0 | 15 | implicit | uni-dir | enabled | 2719752 | | deny,log | any_vrf_any_deny(22) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

border-leaf# contract_parser.py --vrf Prod1:VRF3
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-l4] dst-epg [dst-l4] [flags][contract:{str}] [hit-count]

[9:4134] [vrf:Prod1:VRF3] permit any tn-Prod1/13out-L3Out1/instP-extEpg2(10937) tn-Prod1/13out-L3Out2/instP-extEpg2(24)
[contract:uni/tn-Prod1/brc-vrf1_to_vrf2] [hit=0]
[9:4135] [vrf:Prod1:VRF3] permit any tn-Prod1/13out-L3Out2/instP-extEpg2(24) tn-Prod1/13out-L3Out1/instP-extEpg2(10937)
[contract:uni/tn-Prod1/brc-vrf1_to_vrf2] [hit=0]
[16:4130] [vrf:Prod1:VRF3] permit arp epg:any epg:any [contract:implicit] [hit=0]
[21:4131] [vrf:Prod1:VRF3] deny,log any epg:any epg:any [contract:implicit] [hit=0]
[22:4132] [vrf:Prod1:VRF3] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]
```



# Intra-Fabric forwarding



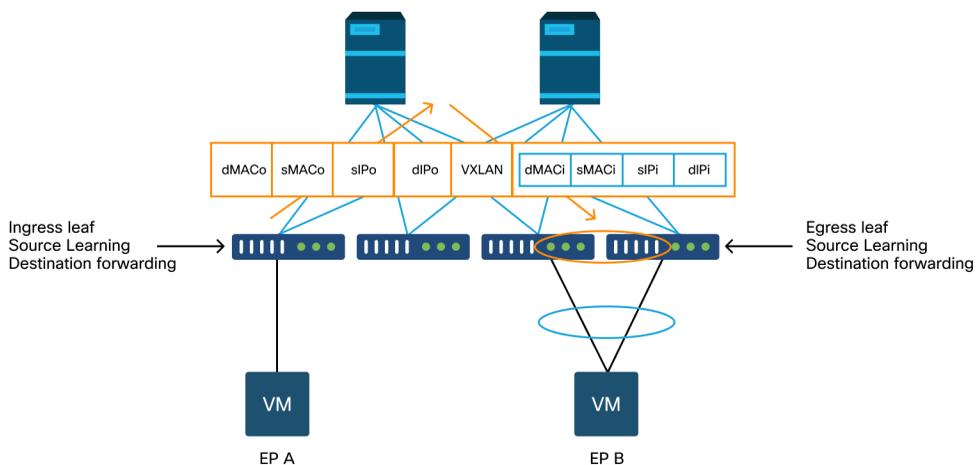
## Overview

This chapter explains a general forwarding path within a Cisco ACI fabric. A few troubleshooting scenarios will be depicted to facilitate the reader in learning how packets flow through the fabric. Typically, this troubleshooting is done to identify packet drops or unexpected behavior of the fabric with regards to endpoint connectivity.

This chapter will not focus on detailed architecture of the forwarding behavior inside of an ACI fabric. For more information about this, refer to other Cisco resources such as the Cisco Live session "BRKACI-3545 Mastering ACI Forwarding Behavior" which can be found in the Cisco Live On-Demand Library.

The figure below illustrates a simple example of one of the forwarding scenarios between an endpoint A and an endpoint B that will be the subject of this chapter.

ACI forwarding behavior overview



In this scenario, the EP A is a VM running on a single-homed host. EP B resides on a host dual-homed to the ACI fabric. Assuming the EPs in the above scenario have not yet established a flow, when a packet flows from the source to the destination leaf, the following happens in the ACI fabric:

- 1 Source leaf learns the source of traffic as an endpoint.
- 2 Source leaf will decide where to send, based on the destination IP or MAC of traffic.
- 3 Source leaf will either:
  - Forward the packet to a local endpoint.
  - Encapsulate the packet with a VXLAN (and subsequent Eth/IP) header and forward it to the destination leaf via a spine.
- 4 Packet is routed in VRF overlay-1 based on TEP IP (or VPC VIP in case of a VPC leaf pair) in the outer header using the IS-IS routing table.
- 5 Packet reaches destination leaf.
  - Destination leaf learns the source EP based on the inner Eth/IP headers.
  - Destination leaf makes forwarding decision based on the destination IP/MAC of the inner header (original packet).
- 6 Packet is sent out of the fabric.

In the case where destination is unknown (i.e. endpoint learning has yet to occur) the fabric will behave differently. More information on this will be covered along with example troubleshooting scenarios in this chapter.

The following troubleshooting scenarios will be detailed in the next chapters:

- Layer 2 forwarding — 2 EPs in the same BD with no unicast routing.
- Unknown Layer 2 unicast traffic — BD in flood mode.

- Unknown Layer 2 unicast traffic – BD in hardware-proxy mode.
- Layer 3 forwarding – EPs in different BDs.
- Layer 3 forwarding – Unknown EP IP.
- ACI Multi-Pod forwarding.
- Intermittent drops.
- Interface drops.

# Tools

In order to troubleshoot a forwarding problem from an ACI perspective, the following needs to be understood:

- 1 Which switch is receiving a flow?
- 2 What forwarding decision is that switch making?
- 3 Is the switch dropping it?

ACI includes several tools which allow the user to gain in-depth insights into what is happening to a specific flow. The next several sections will demonstrate these tools in detail so only a high-level introduction is provided here.

## **SPAN and ERSPAN**

SPAN and ERSPAN are both tools that allow all or some traffic received at a specific location to be replicated to another location. The end device that the replicated traffic is sent to is expected to be running some type of packet sniffer/analyzer application. Traditional SPAN involves replicating traffic that is being received on one port and passing out through another port. ACI supports doing this in addition to ERSPAN.

ERSPAN follows the same concept except replicating the traffic out a local port; the replicated traffic is encapsulated in GRE and sent to a remote destination. In ACI, this ERSPAN destination must only be learned as a Layer 3 endpoint and it can be any EPG in any VRF.

It's usually a good idea to always have SPAN destinations connected to the fabric to minimize preparation time during troubleshooting and allow for rapid ERSPAN session config and capture.

## ELAM

### Overview

Embedded Logic Analyzer Module (ELAM) is a tool that allows a user to set conditions in hardware and capture the first packet or frame that matches the set conditions. A successful capture will cause the ELAM status to show as 'triggered'. Once triggered, the ELAM is disabled and a dump can be collected to analyze the vast number of forwarding decisions that the switch ASIC is making with that packet/frame. ELAM is implemented at the ASIC level and will not impact CPU or other resources on the switch.

The forwarding examples in this book will use ELAM as a means of verifying what is happening with the flow. Examples will show both the leaf CLI version and the ELAM Assistant App.

**This guide will not cover usage of ELAM on first generation leaf switches (switches without EX, FX, or FX2 suffix).**

Before using the tool, it is important to understand the structure of the command syntax.

Example on leaf CLI:

- `vsh_lc` [This command enters the line card shell where ELAMs are run]
- `debug platform internal <asic> elam asic 0` [refer to the ASICs table]

Set Conditions to Trigger

- `trigger reset` [ensures no existing triggers are running]
- `trigger init in-select <number> out-select <number>` [determines what information about a packet is displayed and which conditions can be set]
- `set outer/inner` [sets conditions]

- **start** [starts the trigger]
- **status** [checks if a packet is captured]

Generate the Dump containing the packet analysis

- **ereport** [display detailed forwarding decision for the packet]

'status' should continually be run to view the state of the trigger. Once a packet matching the defined conditions is detected on the ASIC, the output of 'status' will show 'triggered'. Once the ELAM has been triggered, the details of the switch forwarding decisions can be shown with 'ereport'. Prior to ACI version 4.2, 'report' must be used.

## ASICs

Within the ELAM syntax, note that the ASIC must be specified. Since the ASIC is dependent on the switch model, the following table can be used to determine which ASIC to specify:

ASICs table

Switch/Line card Family	ASIC Family
-EX switches/LCs	TAH
-FX(P) switches/LCs	ROC
-FX2 switches/LCs	HEA
C switches (9364C,9332C)	ROC

## ELAM trigger in-select

The other component of the ELAM that must be understood when running from the CLI is the 'in-select'. The 'in-select' defines which headers the packet/frame should have, and which to match on.

For example, a packet coming from a downlink port that is not VXLAN encapsulated would only have outer Layer 2, Layer 3, and Layer 4 headers.

A packet coming from a front-panel (downlink) port that is VXLAN encapsulated (such as Cisco ACI Virtual Edge in VXLAN mode) or coming from an upstream spine would have VXLAN encapsulation. This means it would have potentially both outer and inner Layer 2, Layer 3, and Layer 4 headers.

Below are all the options:

```
leaf1# vsh_lc

module-1# debug platform internal tah elam asic 0
module-1(DBG-elam)# trigger reset
module-1(DBG-elam)# trigger init in-select ?
10 Outer14-inner14-ieth
13 Outer(12|13|14)-inner(12|13|14)-noieth
14 Outer(12(vntag)|13|14)-inner(12|13|14)-ieth
15 Outer(12|13|14)-inner(12|13|14)-ieth
6 Outer12-outer13-outer14
7 Inner12-inner13-inner14
8 Outer12-inner12-ieth
9 Outer13-inner13
```

If 'in-select 6' is selected the only option is to set conditions and display headers from the outer Layer 2, 3, or 4 headers. If 'in-select 14' is selected the only option is to set conditions for and see the details of the outer and inner Layer 2, 3, and 4 headers.

#### Best practices note:

To capture a packet coming with VLAN encapsulation on a downlink port, use 'in-select 6'

To capture a packet with VXLAN encapsulation (either from a spine or from a leaf with VXLAN encapsulation) use 'in-select 14'

### ELAM trigger out-select

The 'out-select' allows some ability to control which lookup results are displayed in the ELAM report. For most practical purposes 'out-select 0' should be used as it contains most information including the 'drop vector' which will tell if the result of the lookup is to drop the packet/frame.

Note that when 'report' instead of 'ereport' or 'report detail' is used to get ELAM results, 'drop vector' only shows up in 'out-select 1'. However, one should always perform 'ereport' or 'report detail' with 'out-select 0'.

### ELAM set conditions

ELAM supports a large amount of Layer 2, 3, and 4 conditions to look for in a packet. Specifying 'inner' vs. 'outer' determines if the condition should be checked in the inner header (VXLAN encapsulated packet) or outer header.

ARP example:

```
set outer arp source-ip-address 10.0.0.1 target-ip-address 10.0.0.2
```

MAC address example:

```
set outer l2 src_mac aaaa.bbbb.cccc dst_mac cccc.bbbb.aaaa
```

IP address in inner header example:

```
set inner ipv4 src_ip 10.0.0.1 dst_ip 10.0.0.2
```

## Viewing the ELAM report

Verify that the ELAM has triggered with **status**:

```
module-1(DBG-elam-insel6)# status
ELAM STATUS
=====
Asic 0 Slice 0 Status Armed
Asic 0 Slice 1 Status Triggered
```

'ereport' can be used to display the result of the ELAM in an easy to understand format. Note that the ELAM report is saved in the '/var/log/dme/log/' folder on the switch. There will be two files for the ELAM under the folder.

- elam\_<timestamp>.txt
- pretty\_elam\_<timestamp>.txt

## Full ELAM example

The following example would capture a non-VXLAN encapsulated traffic (matching on outer header) coming from a downlink port on an -EX switch:

```
module-1# debug platform internal tah elam asic 0
module-1(DBG-elam)# trigger reset
module-1(DBG-elam)# trigger init in-select 6 out-select 0
module-1(DBG-elam-insel6)# set outer ipv4 src_ip 10.0.0.1 dst_ip 10.0.0.2
module-1(DBG-elam-insel6)# start
module-1(DBG-elam-insel6)# status
module-1(DBG-elam-insel6)# ereport
```

## ELAM Assistant application

The troubleshooting examples in this book will also show the usage of the ELAM Assistant app which can be downloaded through the Cisco DC App Center (<https://dcappcenter.cisco.com>). This tool automates the deployment and interpretation of ELAMs through the GUI on the APIC.

The example below shows the deployment of an ELAM matching a specific source and destination IP on node-101 downlink port

## ElamAssistant

The screenshot shows the ElamAssistant interface within a network management system. The top navigation bar includes System, Tenants, Fabric, Virtual Networking, L4-L7 Services, Admin, Operations, Apps, and Integrations. The 'Apps' tab is active, showing 'ElamAssistant' under 'Apps | Faults'. The main interface is titled 'ELAM Assistant' and 'Capture a packet with ELAM (Embedded Logic Analyzer Module)'. On the left, a sidebar lists nodes: node-101 (d-leaf101), node-102 (d-leaf102), node-103 (d-leaf103), node-201 (d-spine201), node-202 (d-spine202), and Unsupported Nodes. The main area is titled 'ELAM PARAMETERS' and contains a form for configuring a capture rule. The form includes a text input for 'Name your capture: (optional)', a table for 'VxLAN (outer) header' with columns for Status, Node, Direction, Source I/F, and Parameters, and two buttons: 'Set ELAM(s)' and 'Check Trigger'. The table shows a single entry with Status 'Not Set', Node 'node-101', Direction 'from frontport', Source I/F 'any', and Parameters 'src ip 10.0.0.1' and 'dst ip 10.0.0.2'. Below the table is a section for 'ELAM Report Parse Result ( report name: )'.

## ElamAssistant – Detail

This screenshot provides a detailed view of the 'ELAM PARAMETERS' section. The title bar reads 'Capture a packet with ELAM (Embedded Logic Analyzer Module)'. The 'ELAM PARAMETERS' section includes a 'Name your capture: (optional)' text input, a table for 'VxLAN (outer) header', and two buttons: 'Set ELAM(s)' and 'Check Trigger'. The table has columns for Status, Node, Direction, Source I/F, and Parameters. A single entry is shown with Status 'Not Set', Node 'node-101', Direction 'from frontport', Source I/F 'any', and Parameters 'src ip 10.0.0.1' and 'dst ip 10.0.0.2'. Below the table is a section for 'ELAM Report Parse Result ( report name: )'.

The ELAM Assistant also allows for easy usage of more complex matching parameters such as the source interface or VXLAN values.

## fTriage

fTriage is an APIC CLI-based tool that is intended to provide end-to-end automation of ELAM configuration and interpretation. The premise of the tool is that a user can define a specific flow as well as the leaf where the flow should start and then the tool will execute ELAMs on each node, one by one, to examine the forwarding flow. It is particularly useful in large topologies where it is unclear which path a packet will take.

fTriage generates a large log file containing the output of each command executed. The name of this file is visible on the first few lines of the fTriage output.

fTriage completion can take up to 15 minutes.

### Examples

Map out the flow for routed communication between 10.0.1.1 and 10.0.2.1 starting on leaf 104:

```
ftriage route -ii LEAF:104 -dip 10.0.2.1 -sip 10.0.1.1
```

Map out a Layer 2 flow starting on leaf 104:

```
ftriage bridge -ii LEAF:104 -dmac 02:02:02:02:02:02
```

Full fTriage help can be seen by running 'ftriage --help' on the APIC.

## Tcpdump

Tcpdump can be leveraged on ACI switches to capture traffic to and from the control-plane. Note that **only control plane traffic** sent to the switch CPU can be observed in a tcpdump capture. Some examples are: routing protocols, LLDP/CDP, LACP, ARP, etc. To capture dataplane (and control plane) traffic please make use of SPAN and/or ELAM.

To capture on the CPU, the interface that should be specified is `kpm_inb`. Most traditional `tcpdump` options and filters are available.

Example to capture ICMP destined to an SVI on the leaf switch:

```
leaf205# tcpdump -ni kpm_inb icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes
20:24:12.921981 IP 10.0.2.100 > 10.0.2.1: ICMP echo request, id 62762, seq 4096, length 64
20:24:12.922059 IP 10.0.2.1 > 10.0.2.100: ICMP echo reply, id 62762, seq 4096, length 64
20:24:13.922064 IP 10.0.2.100 > 10.0.2.1: ICMP echo request, id 62762, seq 4352, length 64
20:24:13.922157 IP 10.0.2.1 > 10.0.2.100: ICMP echo reply, id 62762, seq 4352, length 64
20:24:14.922231 IP 10.0.2.100 > 10.0.2.1: ICMP echo request, id 62762, seq 4608, length 64
20:24:14.922303 IP 10.0.2.1 > 10.0.2.100: ICMP echo reply, id 62762, seq 4608, length 64
```

In addition, the `-w` option allows the `tcpdump` to write the packet capture to a PCAP file so that it can be opened in tools such as Wireshark.

To use `tcpdump` on the `eth0` interface, which is the out-of-band interface on the switch. This is useful to troubleshoot connectivity of any traffic going through the out-of-band physical port of the switch. This would mainly be control plane-based traffic such as SSH, SNMP, etc.

## Enhanced Endpoint Tracker app

This is an app available through the Cisco DC App Center that adds additional functionality and features to endpoint monitoring and management. Its installation is highly recommended for users exhibiting any type of endpoint movement symptoms including intermittent packet loss, full packet loss, or unexpected learn location.

## On-demand Atomic Counters

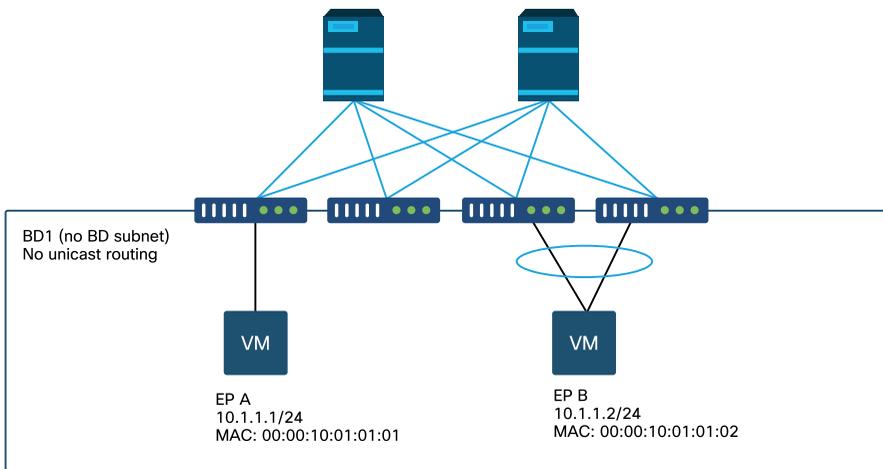
On-demand atomic counters are intended to count packets within a specific flow as they leave on a leaf uplink and are received on another leaf fabric port. They allow some granularity into whether packets were missed or received in excess.

## L2 forwarding: two endpoints in same BD – no unicast routing

This section explains a troubleshooting example where endpoints in the same bridge domain and same subnet can't talk to each other. The figure below illustrates the topology where the BD doesn't have any subnets and has unicast routing disabled.

Typically, when troubleshooting traffic flows with endpoint connectivity, the suggestion is to start identifying a pair of endpoints. Refer to the topology below with EPs A and B. These will respectively have IP addresses 10.1.1.1/24 and 10.1.1.2/24. The MAC addresses will respectively be 00:00:10:01:01:01 and 00:00:10:01:01:02.

Endpoints in same BD and in the same subnet (L2 BD)



In this section there are three scenarios:

- 1 Known Layer 2 unicast flow.
- 2 Unknown Layer 2 unicast flow with BD in flood mode.
- 3 Unknown Layer 2 unicast flow with BD in hardware-proxy mode.

The troubleshooting flows that will be followed can be summarized by the following scheme:

- Level 1 check: GUI validation of the config, faults and endpoints learned.
- Level 2 check: CLI on the leaf switches:
  - Check if the source and destination leaf switches learn the endpoints.
  - Check if spine nodes learn the endpoint in COOP.
- Level 3 check: packet capture:
  - ELAM (ELAM Assistant or CLI) to validate the frame is there.
  - fTriage to track the flow.

## GUI check

The first level of troubleshooting is validating from the GUI that the endpoint MAC was learned properly. This can be done from the operational tab of the EPG where the endpoint sits.

## 'EPG Operational tab &gt; Client End-Points'

								Summary	Policy	Operational	Stats	Health
								Client End-Points	Configured Access Policies	Contracts	Controller End-Points	Deployed Leaves
MAC	IP	Learning Source	Hosting Server	Reporting Controller Name	Interface		Multicast Address	Encap Address				
00:00:10:01:01:01	---	learned	---	---	Pod-1/Node-101/eth1/3 (learned)		---	vlan-2501				
00:00:10:01:01:02	---	learned	---	---	Pod-1/Node-103-104/N3k-3-VPC3-4 (learned)		---	vlan-2501				

Objects Per Page: 15

In this scenario, both endpoints A and B are shown in the GUI. The GUI shows their MAC addresses, the interface where they are connected to the fabric, and the encapsulation – in this case both are in encap VLAN 2501.

It is expected that the IP address isn't learnt from the ACI fabric as the unicast routing has been disabled at the BD level.

Refer to the learning source column in the screenshot above. If it denotes 'learned', the ACI leaf switch received at least one packet from the endpoint.

Since in this case the endpoints are learnt from the ACI fabric, move on to the next troubleshooting case for known Layer 2 unicast traffic.

## Troubleshooting workflow for known Layer 2 unicast traffic

### Ingress leaf source EP MAC learning

In case of Layer 2 forwarding in the same BD, ACI will only learn the source MAC and forward based on the destination MAC. MAC addresses are learnt in the scope of the BD

First, check if the endpoint is learned:

```
leaf1# show endpoint mac 0000.1001.0101
Legend:
s - arp          H - vtep          V - vpc-attached  p - peer-aged
R - peer-attached-r1 B - bounce       S - static        M - span
D - bounce-to-proxy O - peer-attached a - local-aged    m - svc-mgr
L - local       E - shared-service
```

VLAN/ Domain	Encap VLAN	MAC Address IP Address	MAC Info/ IP Info	Interface
4/Prod:VRF1	vlan-2501	0000.1001.0101	L	eth1/3

The above output gives the following information:

- MAC address 0000.1001.0101 is learnt locally (Flag is L for local) on port ethernet 1/3 with encapsulation vlan-2501 in vrf Prod:VRF1.
- Refer to the 'VLAN/Domain' column in the above output. The VLAN ID listed there is the internal VLAN.

### Ingress leaf destination MAC endpoint lookup

Assume the destination MAC is known (known unicast).

```
leaf1# show endpoint mac 0000.1001.0102
Legend:
s - arp                H - vtep                V - vpc-attached        p - peer-aged
R - peer-attached-r1  B - bounce              S - static                M - span
D - bounce-to-proxy   O - peer-attached      a - local-aged           m - svc-mgr
L - local              E - shared-service

-----+-----+-----+-----+-----+
| VLAN/                | Encap                | MAC Address              | MAC Info/              | Interface              |
| Domain              | VLAN                | IP Address              | IP Info                |                       |
+-----+-----+-----+-----+-----+
| 7/Prod:VRF1        | vxlan-16351141     | 0000.1001.0102        |                       | tunnel4                |
+-----+-----+-----+-----+-----+
```

The above output gives the following information:

- MAC address 0000.1001.0102 is not learned locally.
- It is learned from interface tunnel 4.
- It is learned in encapsulation VXLAN-16351141 which corresponds to the BD\_VNID (VXLAN Network ID) of the bridge domain.

Next, check the destination of the tunnel interface using the 'show interface tunnel <x>' command

```
leaf1# show interface tunnel 4
Tunnel4 is up
  MTU 9000 bytes, BW 0 Kbit
  Transport protocol is in VRF "overlay-1"
  Tunnel protocol/transport is vxlan
  Tunnel source 10.0.88.95/32 (lo0)
  Tunnel destination 10.0.96.66
  Last clearing of "show interface" counters never
  Tx
  0 packets output, 1 minute output rate 0 packets/sec
  Rx
  0 packets input, 1 minute input rate 0 packets/sec
```

So, the packet will be encapsulated in VXLAN with source TEP IP 10.0.88.95 (assigned to loopback0) and sent towards the destination TEP IP 10.0.96.66.

Confirm the source IP:

```
leaf1# show ip interface loopback 0 vrf overlay-1
IP Interface Status for VRF "overlay-1"
lo0, Interface status: protocol-up/link-up/admin-up, iod: 4, mode: ptep
IP address: 10.0.88.95, IP subnet: 10.0.88.95/32
IP broadcast address: 255.255.255.255
IP primary address route-preference: 0, tag: 0
```

The destination TEP IP 10.0.96.66 can be one of the following:

- PTEP address of another leaf (can be checked using `acidiag fmvread`)
- VPC VIP (can be seen in 'GUI > Fabric > Access Policies > Policies > Switch > Virtual Port Channel default' (see screenshot below))
- Some loopback IP on a spine switch. Use 'show ip interface vrf overlay-1' command on the spine switch to verify this.

### Explicit VPC Protection Groups

Virtual Port Channel Security Policy - Virtual Port Channel default

Policy    Faults    History

Properties

Explicit VPC Protection Groups:

Name	Domain Policy	Switches	Logical Pair ID	Virtual IP
101-102	default	101, 102	3	10.0.96.67/32
2107-2108		2107, 2108	78	10.2.120.96/32
Pod1-vpc	default	103, 104	1	10.0.96.66/32
pod2-vpc	default	1105, 1106	2	10.1.240.33/32

Show Usage    Reset    Submit

### Ingress leaf switch sending to spine switch

The ingress leaf will now encapsulate the frame into VXLAN with the outer destination IP set to 10.0.96.66 which is the tunnel destination IP listed in the previous 'show interface tunnel 4' command. It will encapsulate it in VXLAN with the VNID of the bridge domain - vxlan-16351141 - as shown in the previous 'show endpoint mac 0000.1001.0102' command output.

Based on the IS-IS route in VRF overlay-1 determine where to send it:

```
leaf1# show ip route 10.0.96.66 vrf overlay-1
IP Route Table for VRF "overlay-1"
'*' denotes best ucast next-hop
'***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.0.96.66/32, ubest/mbest: 4/0
  *via 10.0.88.65, Eth1/49.10, [115/3], 2w5d, isis-isis_infra, isis-l1-int
  *via 10.0.88.94, Eth1/50.128, [115/3], 2w5d, isis-isis_infra, isis-l1-int
```

So, there is ECMP (equal cost multipath) routing to the destination using eth1/49 and 1/50 which are the fabric uplinks to the spine switches.

### Spine forwarding

The VRF overlay-1 routing table on the spine shows that host route 10.0.96.66 is reachable via either to leaf3 or leaf4. This is expected as 10.0.96.66 is the VPC VIP of leaf switches 103 and 104:

```
spine1# show ip route 10.0.96.66 vrf overlay-1
IP Route Table for VRF "overlay-1"
'*' denotes best ucast next-hop
'***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.0.96.66/32, ubest/mbest: 2/0
  *via 10.0.88.91, eth1/3.35, [115/2], 02w05d, isis-isis_infra, isis-l1-int
  *via 10.0.88.90, eth1/4.39, [115/2], 02w05d, isis-isis_infra, isis-l1-int
spine1# show lldp neighbors | egrep "1\|3 |1\|4 "
leaf3      Eth1/3      120      BR      Eth1/49
leaf4      Eth1/4      120      BR      Eth1/49
```

### Egress leaf remote EP MAC learning

In this case, the destination TEP is a VPC pair so the packet will arrive on either leaf3 or leaf4. Refer to the command outputs below. Leaf4 should show similar output. Given they are part of the same VPC pair, all endpoints are synchronized between the two leaf switches.

Endpoint learning for Layer 2 traffic on the egress leaf is based on the source MAC address which is learned in the BD corresponding to the VNID in the received packet. This can be verified in the endpoint table.

The source MAC address lies behind tunnel 26 in VXLAN-16351141.

Tunnel 26 goes to TEP IP 10.0.88.95 which is leaf1:

```
leaf3# show endpoint mac 0000.1001.0101
Legend:
s - arp          H - vtep          V - vpc-attached   p - peer-aged
R - peer-attached-r1 B - bounce       S - static         M - span
D - bounce-to-proxy  O - peer-attached a - local-aged    m - svc-mgr
L - local        E - shared-service

-----+-----+-----+-----+-----+
| VLAN/          | Encap          | MAC Address       | MAC Info/      | Interface      |
| Domain         | VLAN           | IP Address        | IP Info        |                |
+-----+-----+-----+-----+-----+
| 136/Prod:VRF1 | vxlan-16351141 | 0000.1001.0101   |                | tunnel26       |
+-----+-----+-----+-----+-----+

leaf3# show interface tunnel 26
Tunnel26 is up
  MTU 9000 bytes, BW 0 Kbit
  Transport protocol is in VRF "overlay-1"
  Tunnel protocol/transport is vxlan
  Tunnel source 10.0.88.91/32 (100)
  Tunnel destination 10.0.88.95
  Last clearing of "show interface" counters never
  Tx
  0 packets output, 1 minute output rate 0 packets/sec
  Rx
  0 packets input, 1 minute input rate 0 packets/sec

leaf3# acidiag fvnread | egrep "10.0.88.95"
  101      1      leaf1  FD020160TPA  10.0.88.95/32  leaf  active  0
```

### Egress leaf destination MAC lookup

The 'show endpoint' command confirms the destination MAC is learned behind port-channel 1 and uses encapsulation VLAN-2501

```
leaf3# show endpoint mac 0000.1001.0102
Legend:
s - arp          H - vtep          V - vpc-attached  p - peer-aged
R - peer-attached-r1 B - bounce        S - static        M - span
D - bounce-to-proxy O - peer-attached a - local-aged   m - svc-mgr
L - local        E - shared-service
```

VLAN/ Domain	Encap VLAN	MAC Address IP Address	MAC Info/ IP Info	Interface
135/Prod:VRF1	vlan-2501	0000.1001.0102	LpV	po1

This indicates that the frame is leaving the ACI fabric on leaf3 interface port-channel 1 with encap VLAN ID 2501. You can find the BD VNID under the Tenant Operational tab in the GUI.

### Validate both endpoints are learned properly in the spine switch COOP EP repo

The COOP EP repo should be synchronized across all the spine nodes. the COOP EP repo can be checked using the BD VNID as a key and entering the EP MAC address. You can find the BD VNID under the Tenant Operational tab in the GUI.

The source MAC address of this flow is learned from tunnel next-hop 10.0.88.95 which is the TEP IP of leaf1. Additionally, the command output shows VNID 16351141 which corresponds to the correct bridge domain.

```
spine1# show coop internal info repo ep key 16351141 00:00:10:01:01:01
Repo Hdr Checksum : 24197
Repo Hdr record timestamp : 10 01 2019 10:16:50 278195866
Repo Hdr last pub timestamp : 10 01 2019 10:16:50 283699467
Repo Hdr last dampen timestamp : 01 01 1970 00:00:00 0
Repo Hdr dampen penalty : 0
Repo Hdr flags : IN_OBJ EXPORT ACTIVE
EP bd vnid : 16351141 &#10
EP mac : 00:00:10:01:01:01
flags : 0x80
```

```

repo flags : 0x122
Vrf vnid : 2097154
Epg vnid : 0
EVPN Seq no : 0
Remote publish timestamp: 01 01 1970 00:00:00 0
Snapshot timestamp: 10 01 2019 10:16:50 278195866
Tunnel nh : 10.0.88.95
MAC Tunnel : 10.0.88.95
IPv4 Tunnel : 10.0.88.95
IPv6 Tunnel : 10.0.88.95
ETEP Tunnel : 0.0.0.0

```

The destination MAC of this flow is learned against the VPC VIP 10.0.96.66 of leaf3 and leaf4. The EP BD VNID 16351141 is listed as well, which corresponds to the correct BD.

```

spine1# show coop internal info repo ep key 15302583 00:00:10:01:01:02

Repo Hdr Checksum : 16897
Repo Hdr record timestamp : 10 01 2019 11:05:46 351360334
Repo Hdr last pub timestamp : 10 01 2019 11:05:46 352019546
Repo Hdr last dampen timestamp : 01 01 1970 00:00:00 0
Repo Hdr dampen penalty : 0
Repo Hdr flags : IN_OBJ EXPORT ACTIVE
EP bd vnid : 16351141
EP mac : 00:00:10:01:01:02
flags : 0x90
repo flags : 0x122
Vrf vnid : 2097154
Epg vnid : 0
EVPN Seq no : 0
Remote publish timestamp: 01 01 1970 00:00:00 0
Snapshot timestamp: 10 01 2019 11:05:46 351360334
Tunnel nh : 10.0.96.66
MAC Tunnel : 10.0.96.66
IPv4 Tunnel : 10.0.96.66
IPv6 Tunnel : 10.0.96.66
ETEP Tunnel : 0.0.0.0

```

### ELAM output using ELAM Assistant

ELAM Assistant is a powerful ACI App which can simplify the execution of ELAM captures on an ACI fabric.

ELAM Assistant triggers can be started simultaneously on multiple leaf nodes. As a result, specific packets can be checked in parallel in leaf1, leaf3 and leaf4.

The configured ELAM capture will appear as shown below. As observed, the packet is seen on leaf1 (node-101) and leaf3 (node-103).

### ELAM Assistant – parameters

The screenshot shows the 'ELAM ASSISTANT – parameters' interface. At the top, it says 'Capture a packet with ELAM (Embedded Logic Analyzer Module)'. Below this, the 'ELAM PARAMETERS' section is visible. A text input field 'Name your capture:' contains 'L2-only'. The main area is a table with columns: Status, Node, Direction, Source I/F, Parameters, and VxLAN (outer) header.

Status	Node	Direction	Source I/F	Parameters	VxLAN (outer) header
Report Ready	node-101	from frontport	any	src ip: 10.1.1.1 dst ip: 10.1.1.2	
Report Ready	node-103	from SPINE	any	src ip: 10.1.1.1 dst ip: 10.1.1.2	
Set	node-104	from SPINE	any	src ip: 10.1.1.1 dst ip: 10.1.1.2	

At the bottom of the interface, there are buttons for 'Set ELAM' and 'Check Targets'.

The report of leaf1 (node-101) shows the following:

The Captured Packet Information output confirms the packet enters on eth1/3 and has the correct MAC and IP information.

The packet forwarding information shows it's forwarded on eth1/49 to TEP IP 10.0.96.66.

ELAM Assistant – leaf1 (node-101) – Captured Packet Information

Basic Information	
Device Type	LEAF
Packet Direction	ingress (front panel port -> leaf)
Incoming I/F	eth1/3

L2 Header	
Destination MAC	0000.1001.0102
Source MAC	0000.1001.0101
Access Encap VLAN	2501
CoS	0

L3 Header	
L3 Type	IPv4
Destination IP	10.1.1.2
Source IP	10.1.1.1
IP Protocol	0x1 (ICMP)
DSCP	0
TTL	255

No Vx

## ELAM Assistant – leaf1 (node-101) – Packet Forwarding Information

Packet Forwarding Information	
	<b>Forward Result</b>
Destination Type	To another ACI node (or AVS/AVE)
Destination TEP	10.0.96.66 (vPC (103_104))
Destination Physical Port	eth1/49
Sent to SUP/CPU instead	no
SUP Redirect Reason (SUP code)	NONE
	<b>Contract</b>
Destination EPG pcTag (dclass)	32770 (Prod:App:EPG1)
Source EPG pcTag (sclass)	32770 (Prod:App:EPG1)
Contract was applied	1 (Contract was applied on this node)
	<b>Drop</b>

On leaf3 (node-103) on the egress leaf, the following is observed:

In the Captured Packet Information on leaf3, it enters from eth1/49. The outer IP address confirms the following:

- Source TEP: 10.0.88.95
- Destination TEP: 10.0.96.66
- VNID: 16351141 (BD VNID)

ELAM Assistant – leaf3 (node-103) – Captured Packet Information

Captured Packet Information	
Basic Information	
Device Type	LEAF
Packet Direction	egress (spine LC -> leaf)
Incoming I/F	eth1/49

L3 Header (Outer VxLAN)	
L3 Type	IPv4
Destination IP	10.0.96.66 (vPC (103_104))
Source IP	10.0.88.95 (bdsol-aci32-leaf1)
IP Protocol	0x11 (UDP)
DSCP	0
TTL	31
Don't Fragment Bit	0x0 (0x0)

L4 Header (Outer VxLAN)	
L4 Type	iVxLAN
DL (Don't Learn) Bit	0 (not set)
Src Policy Applied Bit	1 (Contract was applied on the previous node)
Dst Policy Applied Bit	1 (Contract was applied on the previous node)
Source EPG (sclass / src pcTag)	0x8002 / 32770 (Prod:App:EPG1)
VRF/BD VNID	15302583 (Prod:BD1)

The Packet Forwarding Information shows the traffic is forwarded on port-channel 1 and specifically ethernet 1/12.

#### ELAM Assistant – leaf3 (node-103) – Packet Forwarding Information

Packet Forwarding Information	
	<b>Forward Result</b>
Destination Type	To a local port
Destination Logical Port	Po1
Destination Physical Port	eth1/12
Sent to SUP/CPU instead	no
SUP Redirect Reason (SUP code)	NONE
	<b>Contract</b>
Destination EPG pcTag (dclass)	32770 (Prod:App:EPG1)
Source EPG pcTag (sclass)	32770 (Prod:App:EPG1)
Contract was applied	1 (Contract was applied on this node)
	<b>Drop</b>
Drop Code	no drop

### Ingress leaf ELAM using CLI

It is recommended to use ELAM Assistant as it simplifies the operation of running ELAM captures. However, it is also possible to use CLI commands on ACI switches to generate an ELAM report. Below is an example of how this would be done.

Use the trigger sequence shown to capture the packet on the ingress leaf. Refer to the "Tools" section for more info regarding ELAM options.

- In this example, the ASIC is 'tah' as the leaf (part number ending '-EX').
- 'in-select 6' is used to capture a packet coming from a downlink port without a VXLAN encap.

- 'out-select 1' ensures the drop vector is also shown (in case of a packet drop).
- The 'reset' command is needed to make sure any previous triggers have been cleaned.
- Even though this is a bridged flow ELAM has visibility into the IP header.
  - As a result, 'ipv4 src\_ip' and 'dst\_ip' can be used to set up the trigger.

```

module-1# debug platform internal tah elam asic 0
module-1(DBG-elam)# trigger init in-select ?
 10 Outer14-inner14-ieth
 13 Outer(12|13|14)-inner(12|13|14)-noieth
 14 Outer(12(vntag)|13|14)-inner(12|13|14)-ieth
 15 Outer(12|13|14)-inner(12|13|14)-ieth
 6  Outer12-outer13-outer14
 7  Inner12-inner13-inner14
 8  Outer12-inner12-ieth
 9  Outer13-inner13

module-1(DBG-elam)# trigger init in-select 6 out-select 1
module-1(DBG-elam-insel6)# reset
module-1(DBG-elam-insel6)# set outer ipv4 src_ip 10.1.1.1 dst_ip 10.1.1.2
module-1(DBG-elam-insel6)# start

```

To see if the packet was received, check the ELAM status. If there is a trigger, that means a packet matching the conditions was caught.

```

module-1(DBG-elam-insel6)# status
ELAM STATUS
=====
Asic 0 Slice 0 Status Triggered
Asic 0 Slice 1 Status Armed

```

The next output shows the report is displayed using the 'ereport' command. The output is very long, so only the beginning is pasted here. But note that the full report is saved for later analysis in a location in the leaf file system. The file name also contains the timestamps when the ELAM was taken.

```
leaf1# ls -al /var/log/dme/log/elam_2019-09-30-03m-23h-14s.txt
-rw-rw-rw- 1 root root 699106 Sep 30 23:03 /var/log/dme/log/elam_2019-09-30-03m-23h-14s.txt
```

The 'ereport' validates the packet has been received and the information is as expected (source and destination MAC, source, and destination IP, etc.)

```
module-1(DBG-elam-insel6)# ereport
Python available. Continue ELAM decode with LC Pkg
ELAM REPORT

=====
                                Trigger/Basic Information
=====
ELAM Report File           : /tmp/logs/elam_2019-09-30-03m-23h-14s.txt
In-Select Trigger         : Outer12-outer13-outer14( 6 )
Out-Select Trigger        : Pktrw-sideband-drpvec( 1 )
ELAM Captured Device      : LEAF
Packet Direction          : ingress
Triggered ASIC type       : Sugarbowl
Triggered ASIC instance   : 0
Triggered Slice           : 0
Incoming Interface        : 0x24( 0x24 )
( Slice Source ID(Ss) in "show plat int hal l2 port gpd" )

=====
                                Captured Packet
=====

Outer Packet Attributes
-----
Outer Packet Attributes    : l2uc ipv4 ip ipuc ipv4uc
Opcode                    : OPCODE_UC

-----
Outer L2 Header
-----
Destination MAC         : 0000.1001.0102
Source MAC             : 0000.1001.0101
802.1Q tag is valid      : yes( 0x1 )
CoS                       : 0( 0x0 )
Access Encap VLAN     : 2501( 0x9C5 )

-----
Outer L3 Header
-----
L3 Type                   : IPv4
IP Version                : 4
DSCP                      : 0
IP Packet Length          : 84 ( = IP header(28 bytes) + IP payload )
Don't Fragment Bit       : not set
```

```

TTL                : 255
IP Protocol Number : ICMP
IP CheckSum        : 51097( 0xC799 )
Destination IP   : 10.1.1.2
Source IP       : 10.1.1.1

```

```

=====
Forwarding Lookup ( FPB )
=====
-----

```

```

Destination MAC (Lookup Key)
-----

```

```

Dst MAC Lookup was performed      : yes
Dst MAC Lookup BD                 : 522( 0x20A )
( Hw BDID in "show plat int hal 12 bd pi" )
Dst MAC Address                   : 0000.1001.0102
-----

```

```

Destination MAC (Lookup Result)
-----

```

```

Dst MAC is Hit                : yes
Dst MAC is Hit Index              : 6443( 0x192B )
( phy_id in "show plat int hal objects ep 12 mac (MAC) extensions" )
or ( HIT IDX in "show plat int hal 13 nexthops" for L3OUT/L3 EP )
.....

```

## Using fTriage to follow the flow

fTriage is run from an APIC CLI and can be used to follow the full path through the ACI fabric. Specify at least the ingress leaf (node-101), the source IP and the destination IP. In this specific case it's a bridged (Layer 2) flow, so the fTriage bridge option is to be used.

Note that fTriage generates a log file in the current directory. This log file will contain all logs and ELAM reports gathered. This allows the packet to be captured at every hop. The short version of the output is below:

```

apic1# ftrriage bridge -ii LEAF:101 -sip 10.1.1.1 -dip 10.1.1.2
fTriage Status: {"dbgFtrriage": {"attributes": {"operState": "InProgress", "pid": "12181", "apicId": "1", "id": "0"}}}
Starting ftrriage
Log file name for the current run is: ftlog_2019-10-01-18-53-24-125.txt
2019-10-01 18:53:24,129 INFO /controller/bin/ftrriage bridge -ii LEAF:101 -sip 10.1.1.1 -dip 10.1.1.2
2019-10-01 18:53:49,280 INFO ftrriage: main:1165 Invoking ftrriage with default password and default
username: apic#fallback\admin
2019-10-01 18:54:10,204 INFO ftrriage: main:839 L2 frame Seen on leaf1 Ingress: Eth1/3 Egress: Eth1/49
Vnid: 15302583

```

```

2019-10-01 18:54:10,422 INFO ftriage: main:242 ingress encap string vlan-2501
2019-10-01 18:54:10,427 INFO ftriage: main:271 Building ingress BD(s), Ctx
2019-10-01 18:54:12,288 INFO ftriage: main:294 Ingress BD(s) Prod:BD1
2019-10-01 18:54:12,288 INFO ftriage: main:301 Ingress Ctx: Prod:VRF1
2019-10-01 18:54:12,397 INFO ftriage: pktrec:490 leaf1: Collecting transient losses snapshot for LC
module: 1
2019-10-01 18:54:30,079 INFO ftriage: main:933 SMAC 00:00:10:01:01:01 DMAC 00:00:10:01:01:02
2019-10-01 18:54:30,080 INFO ftriage: unicast:973 leaf1: <- is ingress node
2019-10-01 18:54:30,320 INFO ftriage: unicast:1215 leaf1: Dst EP is remote
2019-10-01 18:54:31,155 INFO ftriage: misc:659 leaf1: L2 frame getting bridged in SUG
2019-10-01 18:54:31,380 INFO ftriage: misc:657 leaf1: Dst MAC is present in SUG L2 tbl
2019-10-01 18:54:31,826 INFO ftriage: misc:657 leaf1: RwdMAC DIPO(10.0.96.66) is one of dst TEPs
['10.0.96.66']
2019-10-01 18:56:16,249 INFO ftriage: main:622 Found peer-node spine1 and IF: Eth1/1 in candidate list
2019-10-01 18:56:21,346 INFO ftriage: node:643 spine1: Extracted Internal-port GPD Info for lc: 1
2019-10-01 18:56:21,348 INFO ftriage: fcls:4414 spine1: LC trigger ELAM with IFS: Eth1/1 Asic :0 Slice:
0 Srcid: 32
2019-10-01 18:56:54,424 INFO ftriage: main:839 L2 frame Seen on spine1 Ingress: Eth1/1 Egress: LC-1/0
FC-24/0 Port-0 Vnid: 15302583
2019-10-01 18:56:54,424 INFO ftriage: pktrec:490 spine1: Collecting transient losses snapshot for LC
module: 1
2019-10-01 18:57:15,093 INFO ftriage: fib:332 spine1: Transit in spine
2019-10-01 18:57:21,394 INFO ftriage: unicast:1252 spine1: Enter dbg_sub_nexthop with Transit inst: ig
infra: False glbs.dipo: 10.0.96.66
2019-10-01 18:57:21,508 INFO ftriage: unicast:1417 spine1: EP is known in COOP (DIPO = 10.0.96.66)
2019-10-01 18:57:25,537 INFO ftriage: unicast:1458 spine1: Infra route 10.0.96.66 present in RIB
2019-10-01 18:57:25,537 INFO ftriage: node:1331 spine1: Mapped LC interface: LC-1/0 FC-24/0 Port-0 to FC
interface: FC-24/0 LC-1/0 Port-0
2019-10-01 18:57:30,616 INFO ftriage: node:460 spine1: Extracted GPD Info for fc: 24
2019-10-01 18:57:30,617 INFO ftriage: fcls:5748 spine1: FC trigger ELAM with IFS: FC-24/0 LC-1/0 Port-0
Asic :0 Slice: 2 Srcid: 0
2019-10-01 18:57:49,611 INFO ftriage: unicast:1774 L2 frame Seen on FC of node: spine1 with Ingress: FC-
24/0 LC-1/0 Port-0 Egress: FC-24/0 LC-1/0 Port-0 Vnid: 15302583
2019-10-01 18:57:49,611 INFO ftriage: pktrec:487 spine1: Collecting transient losses snapshot for FC
module: 24
2019-10-01 18:57:53,110 INFO ftriage: node:1339 spine1: Mapped FC interface: FC-24/0 LC-1/0 Port-0 to LC
interface: LC-1/0 FC-24/0 Port-0
2019-10-01 18:57:53,111 INFO ftriage: unicast:1474 spine1: Capturing Spine Transit pkt-type L2 frame on
egress LC on Node: spine1 IFS: LC-1/0 FC-24/0 Port-0
2019-10-01 18:57:53,530 INFO ftriage: fcls:4414 spine1: LC trigger ELAM with IFS: LC-1/0 FC-24/0 Port-0
Asic :0 Slice: 0 Srcid: 64
2019-10-01 18:58:26,497 INFO ftriage: unicast:1510 spine1: L2 frame Spine egress Transit pkt Seen on
spine1 Ingress: LC-1/0 FC-24/0 Port-0 Egress: Eth1/3 Vnid: 15302583
2019-10-01 18:58:26,498 INFO ftriage: pktrec:490 spine1: Collecting transient losses snapshot for LC
module: 1
2019-10-01 18:59:28,634 INFO ftriage: main:622 Found peer-node leaf3 and IF: Eth1/49 in candidate list
2019-10-01 18:59:39,235 INFO ftriage: main:839 L2 frame Seen on leaf3 Ingress: Eth1/49 Egress: Eth1/12
(Po1) Vnid: 11364
2019-10-01 18:59:39,350 INFO ftriage: pktrec:490 leaf3: Collecting transient losses snapshot for LC
module: 1
2019-10-01 18:59:54,373 INFO ftriage: main:522 Computed egress encap string vlan-2501
2019-10-01 18:59:54,379 INFO ftriage: main:313 Building egress BD(s), Ctx
2019-10-01 18:59:57,152 INFO ftriage: main:331 Egress Ctx Prod:VRF1
2019-10-01 18:59:57,153 INFO ftriage: main:332 Egress BD(s): Prod:BD1

```

```

2019-10-01 18:59:59,230 INFO ftrriage: unicast:1252 leaf3: Enter dbg_sub_nexthop with Local inst: eg infra:
False glbs.dipo: 10.0.96.66
2019-10-01 18:59:59,231 INFO ftrriage: unicast:1257 leaf3: dbg_sub_nexthop invokes dbg_sub_eg for vip
2019-10-01 18:59:59,231 INFO ftrriage: unicast:1784 leaf3: <- is egress node
2019-10-01 18:59:59,377 INFO ftrriage: unicast:1833 leaf3: Dst EP is local
2019-10-01 18:59:59,378 INFO ftrriage: misc:657 leaf3: EP if(Po1) same as egr if(Po1)
2019-10-01 18:59:59,378 INFO ftrriage: misc:659 leaf3: L2 frame getting bridged in SUG
2019-10-01 18:59:59,613 INFO ftrriage: misc:657 leaf3: Dst MAC is present in SUG L2 tbl
2019-10-01 19:00:06,122 INFO ftrriage: main:961 Packet is Exiting fabric with peer-device: n3k-3 and
peer-port: Ethernet1/16

```

## Troubleshooting workflow for unknown Layer 2 unicast traffic — BD in flood mode

In this example, the destination MAC is unknown. The destination MAC lookup on the ingress leaf shows no output.

```

leaf1# show endpoint mac 0000.1001.0102
Legend:
s - arp                H - vtep                V - vpc-attached        p - peer-aged
R - peer-attached-r1  B - bounce              S - static               M - span
D - bounce-to-proxy   O - peer-attached      a - local-aged          m - svc-mgr
L - local              E - shared-service

```

VLAN/ Domain	Encap VLAN	MAC Address IP Address	MAC Info/ IP Info	Interface

Given the BD is set to 'Flood' for L2 Unknown Unicast, here is what will happen at a high level:

- 1 Ingress leaf will hash the packet header to assign it to one of the FTAGs (from 0 to 15).
- 2 Ingress leaf will encapsulate the frame in a VXLAN packet with the BD VNID. The outer destination IP will be the BD GIPO + FTAG.
- 3 It will be flooded in the fabric following a tree topology and should reach every leaf node that has the BD deployed.

This section will highlight what can be checked.

### Finding BD GIPo

The GUI identifies multicast group 225.1.5.48 used by the BD for multi-destination traffic.

BD GIPo

Bridge Domain - BD1 🔖 ?

Summary Policy Operational Stats Health Faults History

General L3 Configurations Advanced/Troubleshooting

100 🔄 +

Properties

Unknown Unicast Traffic Class ID: 16386  
 Segment: 15302583  
 Multicast Address: 225.1.5.48  
 Monitoring Policy:  ▼  
 First Hop Security Policy:  ▼  
 Optimize WAN Bandwidth:   
 NetFlow Monitor Policies: 🗑️ +

NetFlow IP Filter Type	NetFlow Monitor Policy
No items have been found. Select Actions to create a new item.	

### ELAM – ingress leaf – flooded traffic

Using ELAM Assistant, the ELAM report on the ingress leaf is checked. This shows that the frame was flooded in the BD and is egressing on all fabric uplinks (here eth1/49, 1/50, 1/51 and 1/52).

## ELAM Assistant - ingress leaf - Packet Forwarding Information

## Packet Forwarding Information

Forward Result	
Destination Type	Flood in BD
Destination Ports	eth1/51, eth1/50, eth1/52, eth1/49 (overlay (Fabric uplink))
vPC Designated Forwarder (DF)	yes
Sent to SUP/CPU as well	no
SUP Redirect Reason (SUP code)	NONE

Contract	
Destination EPG pcTag (dclass)	16386 (null)
Source EPG pcTag (sclass)	32770 (null)
Contract was applied	0 (Contract was not applied on this node)

Drop	
Drop Code	no drop

To find the FTAG value selected by the ingress leaf, go to the raw report of the ELAM Assistant.

```
sug_lu2ba_sb_info.mc_info.mc_info_nopad.ftag: 0xC
```

When converting the hexadecimal value of 0xC to decimal, this results in FTAG 12.

### Drawing the FTAG topology

FTAG topology is computed by IS-IS. A tree topology is created for each FTAG value, with a root and output interface list which allows for an optimal load spread topology.

Display the local FTAG topology using the following command. In the example below, we're using FTAG ID 12 topology on spine1.

```
spine1# show isis internal mcast routes ftag
IS-IS process: isis_infra
  VRF : default
  FTAG Routes
  =====
  FTAG ID: 12 [Enabled] Cost:( 2/ 11/ 0)
  -----
  Root port: Ethernet1/4.39
  OIF List:
    Ethernet1/11.11
    Ethernet1/12.12
```

Drawing the full FTAG topology in a large ACI fabric can prove to be a long and complex task. The 'aci-ftag-viewer' Python script (<https://github.com/agccie/aci-ftag-viewer>) can be copied onto an APIC. It generates the complete FTAG topology of the fabric in a single pass.

The output below displays the FTAG 12 tree in Pod1 of a Multi-Pod fabric and includes the FTAG topology across the IPN devices.

This shows that if traffic enters the ACI fabric from leaf101 it will traverse the following paths as listed in the script's output below.

```
admin@apic1:tmp> python aci_ftag_viewer.py --ftag 12 --pod 1
#####
# Pod 1 FTAG 12
# Root spine-204
# active nodes: 8, inactive nodes: 1
#####
spine-204
+- 1/1 ----- 1/52 leaf-101
+- 1/2 ----- 1/52 leaf-102
```

```

+- 1/1 ----- 1/52 leaf-101
+- 1/2 ----- 1/52 leaf-102
+- 1/3 ----- 1/52 leaf-103
+- 1/4 ----- 1/52 leaf-104
      +- 1/49 ----- 1/4 spine-201
      |               +- 1/11 ..... (EXT) Eth2/13 n7706-01-Multipod-A1
      |               +- 1/12 ..... (EXT) Eth2/9 n7706-01-Multipod-A2
      |
      +- 1/50 ----- 1/4 spine-202
      |               +- 1/11 ..... (EXT) Eth2/14 n7706-01-Multipod-A1
      |               +- 1/12 ..... (EXT) Eth2/10 n7706-01-Multipod-A2
      |
      +- 1/51 ----- 2/4 spine-203
      |               +- 2/11 ..... (EXT) Eth2/15 n7706-01-Multipod-A1
      |               +- 2/12 ..... (EXT) Eth2/11 n7706-01-Multipod-A2
+- 1/11 ..... (EXT) Eth2/16 n7706-01-Multipod-A1
+- 1/12 ..... (EXT) Eth2/12 n7706-01-Multipod-A2

```

### ELAM – egress leaf – flooded traffic

In this case, the flooded traffic reaches every leaf in the ACI fabric. So, it will reach both leaf3 and leaf4 which are the VPC pair. Both of those leaf nodes have a VPC to the destination. To avoid duplicate packets, the VPC pair elects only one leaf to forward the flooded traffic to the destination. The elected leaf is called VPC DF leaf (VPC designated forwarder leaf).

This can be checked in ELAM using the following trigger on both leaf nodes.

```

module-1# debug platform internal tah elam asic 0
module-1(DBG-elam)# trigger reset
module-1(DBG-elam)# trigger init in-select 14 out-select 1
module-1(DBG-elam-inse114)# set inner ipv4 src_ip 10.1.1.1 dst_ip 10.1.1.2
module-1(DBG-elam-inse114)# start

```

leaf3 output:

```

module-1(DBG-elam-inse114)# ereport | egrep vpc.*df
sug_lub_latch_results_vec.lub4_1.vpc_df: 0x1

```

leaf4 output:

```
module-1(DBG-elam-inse114)# ereport | egrep vpc.*df
sug_lub_latch_results_vec.lub4_1.vpc_df: 0x0
```

In the above output, leaf3 has value '0x1' set for the 'vpc\_df' field, whereas leaf4 has '0x0' set for the 'vpc\_df' field. Hence the designated forwarder will be leaf3. leaf3 will forward the flooded packet on its VPC link to the destination EP.

## Troubleshooting workflow for unknown Layer 2 unicast traffic — BD in hardware proxy

The current scenario listed is the one for Layer 2 unknown unicast traffic with the BD in hardware proxy mode. In this scenario, given the ingress leaf does not know the destination MAC address, it will forward the packet to the spine anycast proxy-mac address. The spine will perform a COOP lookup for the destination MAC.

If the lookup succeeds as shown below, the spine will rewrite the outer destination IP to the tunnel destination (here 10.0.96.66) and will send it to the leaf3-leaf4 VPC pair.

```
spine1# show coop internal info repo ep key 15302583 00:00:10:01:01:02
```

```
Repo Hdr Checksum : 16897
Repo Hdr record timestamp : 10 01 2019 11:05:46 351360334
Repo Hdr last pub timestamp : 10 01 2019 11:05:46 352019546
Repo Hdr last dampen timestamp : 01 01 1970 00:00:00 0
Repo Hdr dampen penalty : 0
Repo Hdr flags : IN_OBJ EXPORT ACTIVE
EP bd vnid : 16351141
EP mac : 00:00:10:01:01:02
flags : 0x90
repo flags : 0x122
Vrf vnid : 2097154
Epg vnid : 0
EVPN Seq no : 0
Remote publish timestamp: 01 01 1970 00:00:00 0
Snapshot timestamp: 10 01 2019 11:05:46 351360334
Tunnel nh : 10.0.96.66
MAC Tunnel : 10.0.96.66
IPv4 Tunnel : 10.0.96.66
IPv6 Tunnel : 10.0.96.66
ETEP Tunnel : 0.0.0.0
```

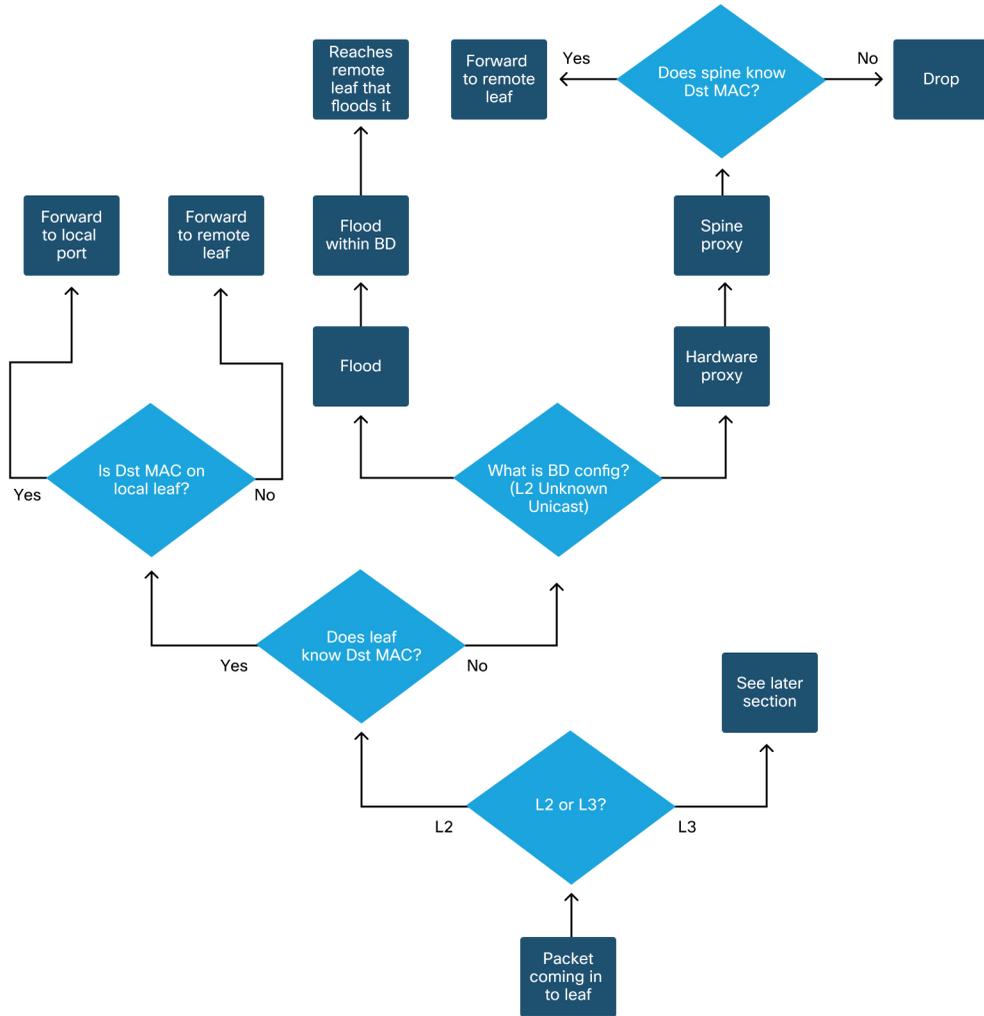
If the lookup fails (endpoint is unknown in the ACI fabric), the spine will drop the unknown unicast.

```
spine1# show coop internal info repo ep key 15302583 00:00:10:01:01:02  
Key not found in repo
```

## Summary

The following diagram summarizes the possible forwarding behavior for Layer 2 traffic in the ACI fabric.

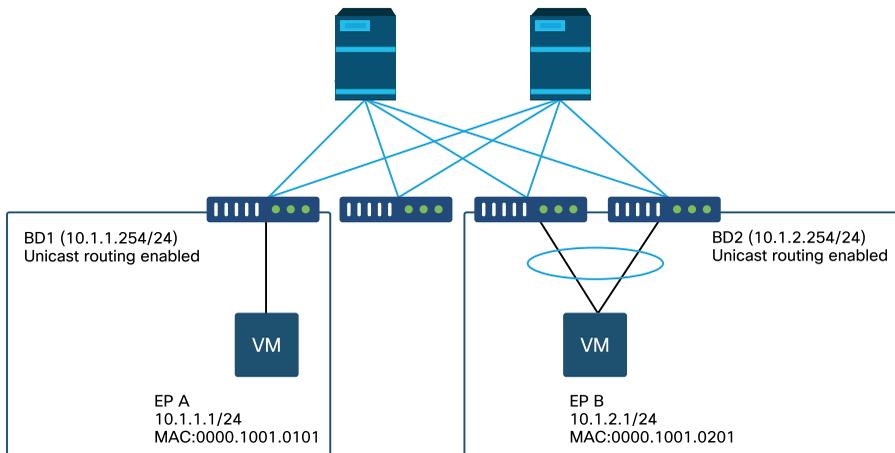
ACI fabric Layer 2 forwarding behavior



## L3 forwarding: two endpoints in different BDs

This chapter explains a troubleshooting example where endpoints in different bridge domains can't talk to each other. This would be a flow routed by ACI fabric. Figure 1 illustrates the topology.

Endpoints in different bridge domains



## High level troubleshooting workflow

The following are typical troubleshooting steps and verification commands:

- First checks – validate programming:
  - BD pervasive gateway should be pushed to leaf nodes.
  - Route to the destination BD subnet should be pushed to leaf nodes.
  - ARP for the default gateway of the hosts should be resolved.
- Second checks – validate learning and table entries via CLI on leaf nodes:
  - Check the source leaf and destination leaf nodes learn the endpoint and whether it learns the destination endpoint:
    - Endpoint table – 'show endpoint'.
    - TEP destination – 'show interface tunnel <x>'.
    - Locating TEP destination in 'show ip route <TEP address> vrf overlay-1' command.
  - Check spine nodes learns the endpoint:
    - 'show coop internal info'.
- Third checks – grab a packet and analyze the forwarding decisions:
  - With ELAM (ELAM Assistant or CLI) to validate the frame is there.
  - Or with fTriage to track the flow.

## Troubleshooting workflow for known endpoints

### Check the pervasive gateway of the BD

In this example, the following source and destination endpoints will be used:

- EP A 10.1.1.1 under leaf1.
- EP B 10.1.2.1 under VPC pair leaf3 and leaf4.

Following pervasive gateways should be seen:

- 10.1.1.254/24 for BD1 gateway on leaf1.
- 10.1.2.254/24 for BD2 gateway on leaf3 and leaf4.

This can be checked using: 'show ip interface vrf <vrf name>' on the leaf nodes.

leaf1:

```
leaf1# show ip interface vrf Prod:VRF1
IP Interface Status for VRF "Prod:VRF1"
vlan7, Interface status: protocol-up/link-up/admin-up, iod: 106, mode: pervasive
IP address: 10.1.1.254, IP subnet: 10.1.1.0/24
IP broadcast address: 255.255.255.255
IP primary address route-preference: 0, tag: 0
```

leaf3 and 4:

```
leaf3# show ip interface vrf Prod:VRF1
IP Interface Status for VRF "Prod:VRF1"
vlan1, Interface status: protocol-up/link-up/admin-up, iod: 159, mode: pervasive
IP address: 10.1.2.254, IP subnet: 10.1.2.0/24
IP broadcast address: 255.255.255.255
IP primary address route-preference: 0, tag: 0
```

```
leaf4# show ip interface vrf Prod:VRF1
IP Interface Status for VRF "Prod:VRF1"
vlan132, Interface status: protocol-up/link-up/admin-up, iod: 159, mode: pervasive
IP address: 10.1.2.254, IP subnet: 10.1.2.0/24
```

```
IP broadcast address: 255.255.255.255
IP primary address route-preference: 0, tag: 0
```

Note that leaf3 and leaf4 have the same pervasive gateway address, but different VLAN encapsulation for the SVI will likely be seen.

- leaf3 uses VLAN 1.
- leaf4 uses VLAN 132.

This is expected as VLAN 1 or VLAN 132 is local VLAN on the leaf.

If the pervasive gateway IP address is not pushed to the leaf, verify in APIC GUI that there are no faults that would prevent the VLAN from being deployed.

### Checking routing table on the leaf

Leaf1 does not have any endpoint in subnet 10.1.2.0/24, however it must have the route to that subnet in order to reach it:

```
leaf1# show ip route 10.1.2.0/24 vrf Prod:VRF1
IP Route Table for VRF "Prod:VRF1"
'*' denotes best ucast next-hop
'***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
10.1.2.0/24, ubest/mbest: 1/0, attached, direct, pervasive
  *via 10.0.8.65%overlay-1, [1/0], 00:22:37, static, tag 4294967294
    recursive next hop: 10.0.8.65/32%overlay-1
```

Note that the route flagged with 'pervasive' and 'direct' have next-hop of 10.0.8.65. This is the anycast-v4 loopback address which exists on all spines.

```
leaf1# show isis dtaps vrf overlay-1 | egrep 10.0.8.65
10.0.8.65      SPINE    N/A      PHYSICAL, PROXY-ACAST-V4
bdsol-aci32-leaf1#
```

Similarly, leaf3 and leaf4 should have route for 10.1.1.0/24.

```
leaf3# show ip route 10.1.1.1 vrf Prod:VRF1
IP Route Table for VRF "Prod:VRF1"
'*' denotes best ucast next-hop
 '**' denotes best mcast next-hop
 '[x/y]' denotes [preference/metric]
 '%<string>' in via output denotes VRF <string>
10.1.1.0/24, ubest/mbest: 1/0, attached, direct, pervasive
   *via 10.0.8.65%overlay-1, [1/0], 00:30:25, static, tag 4294967294
     recursive next hop: 10.0.8.65/32%overlay-1
```

If these routes are missing, it is likely because there is no contract between an EPG in BD1 and an EPG in BD2. If there is no local endpoint in BD1 under a leaf, the BD1 pervasive gateway doesn't get pushed to the leaf. If there is a local endpoint in an EPG that has a contract with another EPG in BD1, the BD1 subnet gets learned on the leaf.

### ARP resolution for the default gateway IP

Since the leaf where a local endpoint resides should have a pervasive gateway, ARP requests for the pervasive gateway should always be resolved by the local leaf. This can be checked on the local leaf using the following command:

```
leaf1# show ip arp internal event-history event | egrep 10.1.1.1
 [116] TID 26571:arp_handle_arp_request:6135: log_collect_arp_pkt; sip = 10.1.1.1; dip = 10.1.1.254;interface
 = Vlan7; phy_interface = Ethernet1/3; flood = 0; Info = Sent ARP response.
 [116] TID 26571:arp_process_receive_packet_msg:8384: log_collect_arp_pkt; sip = 10.1.1.1; dip =
 10.1.1.254;interface = Vlan7; phy_interface = Ethernet1/3;Info = Received arp request
```

### Ingress leaf source IP and MAC endpoint learning

In case of Layer 3 forwarding, ACI will perform Layer 3 source IP learning and destination IP lookup. Learned IP addresses are scoped to the VRF.

This can be checked on the GUI in an EPG's 'operational' tab. Note that here the IP and the MAC are both learned.

## EPG Operational End-Points

EPG - EPG1

Summary Policy **Operational** Stats Health Faults History

Client End-Points Configured Access Policies Contracts Controller End-Points Deployed Leaves Learned End-Points

End Point	MAC	IP	Learning Source	Hosting Server	Reportin Controll Name	Interface	Multicas Encap Address
EP-00:00:10:01:01:01	00:00:10:01:01:01	10.1.1.1	learned	---	---	Pod-1/Node-101/eth1/3 (learned)	vlan-2501
EP-00:00:10:01:01:02	00:00:10:01:01:02	10.1.1.2...	learned	---	---	Pod-1/Node-103-104/N3k-3-VPC3-4 (learned)	vlan-2501

## EPG Operational End-Points – detail

End Point	MAC	IP	Learning Source	Hosting Server	Reportin Controll Name	Interface	Multicas Encap Address
EP-00:00:10:01:01:01	00:00:10:01:01:01	10.1.1.1	learned	---	---	Pod-1/Node-101/eth1/3 (learned)	vlan-2501
EP-00:00:10:01:01:02	00:00:10:01:01:02	10.1.1.2,...	learned	---	---	Pod-1/Node-103-104/N3k-3-VPC3-4 (learned)	vlan-2501

Check local endpoint is learned on the local leaf. Here check on leaf1 that IP 10.1.1.1 is learned:

```
leaf1# show endpoint ip 10.1.1.1
```

Legend:

```
s - arp          H - vtep          V - vpc-attached    p - peer-aged
R - peer-attached-r1 B - bounce        S - static          M - span
D - bounce-to-proxy O - peer-attached  a - local-aged     m - svc-mgr
L - local        E - shared-service
```

```
+-----+-----+-----+-----+-----+
| VLAN/ | Encap | MAC Address | MAC Info/ | Interface |
| Domain | VLAN  | IP Address  | IP Info   |            |
+-----+-----+-----+-----+-----+
| 46    | vlan-2501 | 0000.1001.0101 | L         | eth1/3    |
| Prod:VRF1 |          | 10.1.1.1 | L         | eth1/3    |
+-----+-----+-----+-----+-----+
```

As shown above, the endpoint content is:

- BD (internal VLAN for BD is 46) with VLAN encapsulation of the EPG (vlan-2501) and the MAC address learned on eth1/3
- VRF (Prod:VRF1) with the IP 10.1.1.1



```
leaf1# show interface tunnel 4
Tunnel4 is up
  MTU 9000 bytes, BW 0 Kbit
  Transport protocol is in VRF "overlay-1"
  Tunnel protocol/transport is ipvlan
  Tunnel source 10.0.88.95/32 (lo0)
  Tunnel destination 10.0.96.66
  Last clearing of "show interface" counters never
  Tx
  0 packets output, 1 minute output rate 0 packets/sec
  Rx
  0 packets input, 1 minute input rate 0 packets/sec
```

The destination TEP is the anycast TEP of the leaf3 and 4 VPC pair and is learned via uplinks to spine.

```
leaf1# show ip route 10.0.96.66 vrf overlay-1
IP Route Table for VRF "overlay-1"
'*' denotes best ucast next-hop
'***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
10.0.96.66/32, ubest/mbest: 4/0
  *via 10.0.88.65, eth1/49.10, [115/3], 02w06d, isis-isis_infra, isis-l1-int
  *via 10.0.128.64, eth1/51.8, [115/3], 02w06d, isis-isis_infra, isis-l1-int
  *via 10.0.88.64, eth1/52.126, [115/3], 02w06d, isis-isis_infra, isis-l1-int
  *via 10.0.88.94, eth1/50.128, [115/3], 02w06d, isis-isis_infra, isis-l1-int
```

Additional endpoint information for IP 10.1.2.1 can be collected using the 'show system internal epm endpoint ip <ip>' command.

```
leaf1# show system internal epm endpoint ip 10.1.2.1
MAC : 0000.0000.0000 ::: Num IPs : 1
IP# 0 : 10.1.2.1 ::: IP# 0 flags : ::: 13-sw-hit: No
Vlan id : 0 ::: Vlan vnid : 0 ::: VRF name : Prod:VRF1
BD vnid : 0 ::: VRF vnid : 2097154
Phy If : 0 ::: Tunnel If : 0x18010004
Interface : Tunnel4
Flags : 0x80004420 ::: sclass : 32771 ::: Ref count : 3
EP Create Timestamp : 10/01/2019 13:53:16.228319
EP Update Timestamp : 10/01/2019 14:04:40.757229
EP Flags : peer-aged|IP|sclass|timer|
:::
```

In that output check:

- VRF VNID is populated – this is the VNID used to encapsulate the frame in VXLAN to the fabric.
- MAC address is 0000.0000.0000 as MAC address is never populated on a remote IP entry.
- BD VNID is unknown as for routed frames, the ingress leaf acts as the router and does a MAC rewrite. This means the remote leaf will not have visibility into the BD of the destination, only the VRF.

The frame will now be encapsulated in a VXLAN frame going to the remote TEP 10.0.96.66 with a VXLAN id of 2097154 which is the VNID of the VRF. It will be routed in the overlay-1 routing table (IS-IS route) and will reach the destination TEP. Here it will reach either leaf3 or leaf4 as 10.0.96.66 is the anycast TEP address of the leaf3 and leaf4 VPC pair.

### Source IP learning on egress leaf

The outputs here are taken from leaf3 but would be similar on leaf4. When packets reach leaf3 (destination leaf and owner of the TEP), leaf will learn source IP of the packet in the VRF.

```
leaf3# show endpoint ip 10.1.1.1
```

```
Legend:
```

```
s - arp          H - vtep          V - vpc-attached   p - peer-aged
R - peer-attached-r1 B - bounce       S - static         M - span
D - bounce-to-proxy O - peer-attached a - local-aged    m - svc-mgr
L - local        E - shared-service
```

VLAN/ Domain	Encap VLAN	MAC Address IP Address	MAC Info/ IP Info	Interface
Prod:VRF1		10.1.1.1 p		tunnel26

```
leaf3# show interface tunnel 26
```

```
Tunnel26 is up
```

```
MTU 9000 bytes, BW 0 Kbit
```

```
Transport protocol is in VRF "overlay-1"
```

```
Tunnel protocol/transport is vxlan
```

```
Tunnel source 10.0.88.91/32 (lo0)
Tunnel destination 10.0.88.95
Last clearing of "show interface" counters never
Tx
0 packets output, 1 minute output rate 0 packets/sec
Rx
0 packets input, 1 minute input rate 0 packets/sec
The destination TEP 10.0.88.95 is the TEP address of leaf1 and is learned via all uplinks to spine.
```

### Destination IP lookup on egress leaf

The last step is for the egress leaf to lookup the destination IP. Look at the endpoint table for 10.1.2.1.

This gives the following information:

- The egress leaf knows the destination 10.1.2.1 (similar to a /32 host route in routing table) and the route is learned in correct VRF.
- The egress leaf knows the MAC 0000.1001.0201 (endpoint info).
- The egress leaf knows the traffic destined to 10.1.2.1 must be encapsulated in vlan-2502 and send out on port-channel 1 (po1).

```
leaf3# show endpoint ip 10.1.2.1
Legend:
s - arp          H - vtep          V - vpc-attached  p - peer-aged
R - peer-attached-r1 B - bounce      S - static        M - span
D - bounce-to-proxy O - peer-attached a - local-aged   m - svc-mgr
L - local        E - shared-service
```

VLAN/ Domain	Encap VLAN	MAC Address IP Address	MAC Info/ IP Info	Interface
2	vlan-2502	0000.1001.0201	LpV	po1
Prod:VRF1	vlan-2502	10.1.2.1	LpV	po1

### fTriage to follow the datapath

Use fTriage in the APIC to follow the datapath flow. Remember, fTriage relies on ELAM, so it needs real data flow. This allows confirmation of the full datapath, with confirmation that the packet exits the fabric on leaf3 port 1/16.

```

apic1# ftriage route -ii LEAF:101 -sip 10.1.1.1 -dip 10.1.2.1
fTriage Status: {"dbgFtriage": {"attributes": {"operState": "InProgress", "pid": "6888", "apicId": "1", "id": "0"}}}
Starting ftriage
Log file name for the current run is: ftlog_2019-10-01-21-17-54-175.txt
2019-10-01 21:17:54,179 INFO /controller/bin/ftriage route -ii LEAF:101 -sip 10.1.1.1 -dip 10.1.2.1
2019-10-01 21:18:18,149 INFO ftriage: main:1165 Invoking ftriage with default password and default username:
apic#fallback\admin
2019-10-01 21:18:39,194 INFO ftriage: main:839 L3 packet Seen on bdsol-aci32-leaf1 Ingress: Eth1/3 Egress: Eth1/51 Vnid:
2097154
2019-10-01 21:18:39,413 INFO ftriage: main:242 ingress encap string vlan-2501
2019-10-01 21:18:39,419 INFO ftriage: main:271 Building ingress BD(s), Ctx
2019-10-01 21:18:41,240 INFO ftriage: main:294 Ingress BD(s) Prod:BD1
2019-10-01 21:18:41,240 INFO ftriage: main:301 Ingress Ctx: Prod:VRF1
2019-10-01 21:18:41,349 INFO ftriage: pktrec:490 bdsol-aci32-leaf1: Collecting transient losses snapshot for LC module: 1
2019-10-01 21:19:05,747 INFO ftriage: main:933 SIP 10.1.1.1 DIP 10.1.2.1
2019-10-01 21:19:05,749 INFO ftriage: unicast:973 bdsol-aci32-leaf1: <- is ingress node
2019-10-01 21:19:08,459 INFO ftriage: unicast:1215 bdsol-aci32-leaf1: Dst EP is remote
2019-10-01 21:19:09,984 INFO ftriage: misc:657 bdsol-aci32-leaf1: DMAC(00:22:BD:F8:19:FF) same as RMAC(00:22:BD:F8:19:FF)
2019-10-01 21:19:09,984 INFO ftriage: misc:659 bdsol-aci32-leaf1: L3 packet getting routed/bounced in SUG
2019-10-01 21:19:10,248 INFO ftriage: misc:657 bdsol-aci32-leaf1: Dst IP is present in SUG L3 tbl
2019-10-01 21:19:10,609 INFO ftriage: misc:657 bdsol-aci32-leaf1: RvDMAC DIPo(10.0.96.66) is one of dst TEPs
['10.0.96.66']
2019-10-01 21:20:56,148 INFO ftriage: main:622 Found peer-node bdsol-aci32-spine3 and IF: Eth2/1 in candidate list
2019-10-01 21:21:01,245 INFO ftriage: node:643 bdsol-aci32-spine3: Extracted Internal-port GPD Info for lc: 2
2019-10-01 21:21:01,245 INFO ftriage: fcls:4414 bdsol-aci32-spine3: LC trigger ELAM with IFS: Eth2/1 Asic :0 Slice: 0
Srcid: 32
2019-10-01 21:21:33,894 INFO ftriage: main:839 L3 packet Seen on bdsol-aci32-spine3 Ingress: Eth2/1 Egress: LC-2/0 FC-
22/0 Port-1 Vnid: 2097154
2019-10-01 21:21:33,895 INFO ftriage: pktrec:490 bdsol-aci32-spine3: Collecting transient losses snapshot for LC module: 2
2019-10-01 21:21:54,487 INFO ftriage: fib:332 bdsol-aci32-spine3: Transit in spine
2019-10-01 21:22:01,568 INFO ftriage: unicast:1252 bdsol-aci32-spine3: Enter dbg_sub.nextthop with Transit inst: ig infra:
False glbs.dipo: 10.0.96.66
2019-10-01 21:22:01,682 INFO ftriage: unicast:1417 bdsol-aci32-spine3: EP is known in COOP (DIPo = 10.0.96.66)
2019-10-01 21:22:05,713 INFO ftriage: unicast:1458 bdsol-aci32-spine3: Infra route 10.0.96.66 present in RIB
2019-10-01 21:22:05,713 INFO ftriage: node:1331 bdsol-aci32-spine3: Mapped LC interface: LC-2/0 FC-22/0 Port-1 to FC
interface: FC-22/0 LC-2/0 Port-1
2019-10-01 21:22:10,799 INFO ftriage: node:460 bdsol-aci32-spine3: Extracted GPD Info for fc: 22
2019-10-01 21:22:10,799 INFO ftriage: fcls:5748 bdsol-aci32-spine3: FC trigger ELAM with IFS: FC-22/0 LC-2/0 Port-1 Asic
:0 Slice: 2 Srcid: 24
2019-10-01 21:22:29,322 INFO ftriage: unicast:1774 L3 packet Seen on FC of node: bdsol-aci32-spine3 with Ingress: FC-22/0
LC-2/0 Port-1 Egress: FC-22/0 LC-2/0 Port-1 Vnid: 2097154
2019-10-01 21:22:29,322 INFO ftriage: pktrec:487 bdsol-aci32-spine3: Collecting transient losses snapshot for FC module: 22
2019-10-01 21:22:31,571 INFO ftriage: node:1339 bdsol-aci32-spine3: Mapped FC interface: FC-22/0 LC-2/0 Port-1 to LC
interface: LC-2/0 FC-22/0 Port-1
2019-10-01 21:22:31,572 INFO ftriage: unicast:1474 bdsol-aci32-spine3: Capturing Spine Transit pkt-type L3 packet on egress
LC on Node: bdsol-aci32-spine3 IFS: LC-2/0 FC-22/0 Port-1
2019-10-01 21:22:31,991 INFO ftriage: fcls:4414 bdsol-aci32-spine3: LC trigger ELAM with IFS: LC-2/0 FC-22/0 Port-1 Asic
:0 Slice: 1 Srcid: 0

```

```

2019-10-01 21:22:48,952 INFO ftriage: unicast:1510 bdsol-aci32-spine3: L3 packet Spine egress Transit pkt Seen on bdsol-aci32-spine3 Ingress: LC-2/0 FC-22/0 Port-1 Egress: Eth2/3 Vnid: 2097154
2019-10-01 21:22:48,952 INFO ftriage: pktrec:490 bdsol-aci32-spine3: Collecting transient losses snapshot for LC module: 2
2019-10-01 21:23:50,748 INFO ftriage: main:622 Found peer-node bdsol-aci32-leaf3 and IF: Eth1/51 in candidate list
2019-10-01 21:24:05,313 INFO ftriage: main:839 L3 packet Seen on bdsol-aci32-leaf3 Ingress: Eth1/51 Egress: Eth1/12 (Po1) Vnid: 11365
2019-10-01 21:24:05,427 INFO ftriage: pktrec:490 bdsol-aci32-leaf3: Collecting transient losses snapshot for LC module: 1
2019-10-01 21:24:24,369 INFO ftriage: nxos:1404 bdsol-aci32-leaf3: nxos matching rule id:4326 scope:34 filter:65534
2019-10-01 21:24:25,698 INFO ftriage: main:522 Computed egress encap string vlan-2502
2019-10-01 21:24:25,704 INFO ftriage: main:313 Building egress BD(s), Ctx
2019-10-01 21:24:27,510 INFO ftriage: main:331 Egress Ctx Prod:VRF1
2019-10-01 21:24:27,510 INFO ftriage: main:332 Egress BD(s): Prod:BD2
2019-10-01 21:24:30,536 INFO ftriage: unicast:1252 bdsol-aci32-leaf3: Enter dbg_sub_nextthop with Local inst: eg infra: False glbs.dipo: 10.0.96.66
2019-10-01 21:24:30,537 INFO ftriage: unicast:1257 bdsol-aci32-leaf3: dbg_sub_nextthop invokes dbg_sub_eg for vip
2019-10-01 21:24:30,537 INFO ftriage: unicast:1784 bdsol-aci32-leaf3: <- is egress node
2019-10-01 21:24:30,684 INFO ftriage: unicast:1833 bdsol-aci32-leaf3: Dst EP is local
2019-10-01 21:24:30,685 INFO ftriage: misc:657 bdsol-aci32-leaf3: EP if(Po1) same as egr if(Po1)
2019-10-01 21:24:30,943 INFO ftriage: misc:657 bdsol-aci32-leaf3: Dst IP is present in SUG L3 tbl
2019-10-01 21:24:31,242 INFO ftriage: misc:657 bdsol-aci32-leaf3: RW seg_id:11365 in SUG same as EP segid:11365
2019-10-01 21:24:37,631 INFO ftriage: main:961 Packet is Exiting fabric with peer-device: bdsol-aci32-n3k-3 and peer-port: Ethernet1/12

```

## Packet capture on egress leaf using ELAM Assistant app

Below is the packet captured with the ELAM Assistant app on leaf3 coming from the spine. This shows that:

- The VNID from the outer Layer 4 information (VNID is 2097154).
- Outer L3 header source TEP and destination TEP.

ELAM Assistant – L3 flow egress leaf (part 1)

Device Type	LEAF
Packet Direction	egress (spine LC -> leaf)
Incoming I/F	eth1/51

L2 Header	
Destination MAC	000C.0C0C.0C0C
Source MAC	000C.0C0C.0C0C
Access Encap VLAN	No VLAN Tag
CoS	No VLAN Tag (= No CoS)

L3 Header	
L3 Type	IPv4
Destination IP	10.1.2.1
Source IP	10.1.1.1
IP Protocol	0x1 (ICMP)
DSCP	0
TTL	254
Don't Fragment Bit	0x0 (0x0)
IP Checksum	Unsupported for ELAM with VxLAN data
IP Packet Length	Unsupported for ELAM with VxLAN data

## ELAM Assistant – L3 flow egress leaf (part 2)

<b>L2 Header (Outer VxLAN)</b>	
Destination MAC	000C.0C0C.0C0C
Source MAC	000D.0D0D.0D0D
Access Encap VLAN	2
CoS	0

<b>L3 Header (Outer VxLAN)</b>	
L3 Type	IPv4
Destination IP	10.0.96.66 (vPC (103_104))
Source IP	10.0.88.95 (bdsol-aci32-leaf1)
IP Protocol	0x11 (UDP)
DSCP	0
TTL	31
Don't Fragment Bit	0x0 (0x0)

<b>L4 Header (Outer VxLAN)</b>	
L4 Type	iVxLAN
DL (Don't Learn) Bit	0 (not set)
Src Policy Applied Bit	0 (Contract has yet to be applied)
Dst Policy Applied Bit	0 (Contract has yet to be applied)
Source EPG (sclass / src pcTag)	0x8002 / 32770 (Prod:App:EPG1)
VRF/BD VNID	2097154 (Prod:VRF1)

The Packet Forwarding Information section proves it got out on port-channel 1

### ELAM Assistant – L3 egress leaf – Packet Forwarding Information

Packet Forwarding Information	
	<b>Forward Result</b>
Destination Type	To a local port
Destination Logical Port	Po1
Destination Physical Port	eth1/12
Sent to SUP/CPU instead	no
SUP Redirect Reason (SUP code)	NONE
	<b>Contract</b>
Destination EPG pcTag (dclass)	32771 (null)
Source EPG pcTag (sclass)	32770 (null)
Contract was applied	1 (Contract was applied on this node)
	<b>Drop</b>
Drop Code	no drop

## Troubleshooting workflow for unknown endpoints

This section shows what differs when the ingress leaf does not know the destination IP.

### Ingress leaf destination IP lookup

The first step is to check if there is an endpoint learn for the destination IP.

```
leaf1# show endpoint ip 10.1.2.1
Legend:
s - arp           H - vtep           V - vpc-attached   p - peer-aged
R - peer-attached-r1 B - bounce         S - static          M - span
D - bounce-to-proxy 0 - peer-attached a - local-aged     m - svc-mgr
L - local         E - shared-service
```

VLAN/ Domain	Encap VLAN	MAC Address IP Address	MAC Info/ IP Info	Interface	
					<NO ENTRY>

There is nothing in endpoint table for the destination, so next step is to check the routing table looking for the longest prefix match route to the destination:

```
leaf1# show ip route 10.1.2.1 vrf Prod:VRF1
IP Route Table for VRF "Prod:VRF1"
**' denotes best ucast next-hop
***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.1.2.0/24, ubest/mbest: 1/0, attached, direct, pervasive
 *via 10.0.8.65%overlay-1, [1/0], 01:40:18, static, tag 4294967294
  recursive next hop: 10.0.8.65/32%overlay-1
```

Falling on the /24 BD subnet 10.1.2.0/24 means the leaf will encapsulate the frame in VXLAN with destination TEP 10.0.8.65 (anycast-v4 on spine). The frame will use a VXLAN id which is the VRF VNID.

### COOP lookup on spine – destination IP is known

The packet will reach one of the spines that does COOP lookup in the IP database. The source must be verified and the destination IP needs to be learned correctly from the COOP database.

To find an IP in the COOP database, the key is VRF VNID (2097154 in this example)

From the output below, there is confirmation that the COOP database has the entry for the source IP from TEP 10.0.88.95 (leaf1) correctly.

```
spine1# show coop internal info ip-db key 2097154 10.1.1.1
IP address : 10.1.1.1
Vrf : 2097154
Flags : 0
EP bd vnid : 15302583
EP mac : 00:00:10:01:01:01
Publisher Id : 10.0.88.95
Record timestamp : 10 01 2019 14:16:50 522482647
```

```

Publish timestamp : 10 01 2019 14:16:50 532239332
Seq No: 0
Remote publish timestamp: 01 01 1970 00:00:00 0
URIB Tunnel Info
Num tunnels : 1
    Tunnel address : 10.0.88.95
    Tunnel ref count : 1

```

The output below shows that the COOP database has the entry for the destination IP from TEP 10.0.96.66 (Anycast TEP of the leaf3 and 4 VPC pair) correctly

```

spine1# show coop internal info ip-db key 2097154 10.1.2.1
IP address : 10.1.2.1
Vrf : 2097154
Flags : 0
EP bd vnid : 15957974
EP mac : 00:00:10:01:02:01
Publisher Id : 10.0.88.90
Record timestamp : 10 01 2019 14:52:52 558812544
Publish timestamp : 10 01 2019 14:52:52 559479076
Seq No: 0
Remote publish timestamp: 01 01 1970 00:00:00 0
URIB Tunnel Info
Num tunnels : 1
    Tunnel address : 10.0.96.66
    Tunnel ref count : 1

```

In the scenario here, COOP knows the destination IP so it will rewrite the destination IP of the outer IP header in the VXLAN packet to be 10.0.96.66 and then will send to leaf3 or leaf4 (depending on ECMP hashing). Note that the source IP of the VXLAN frame is not changed so it is still the leaf1 PTEP.

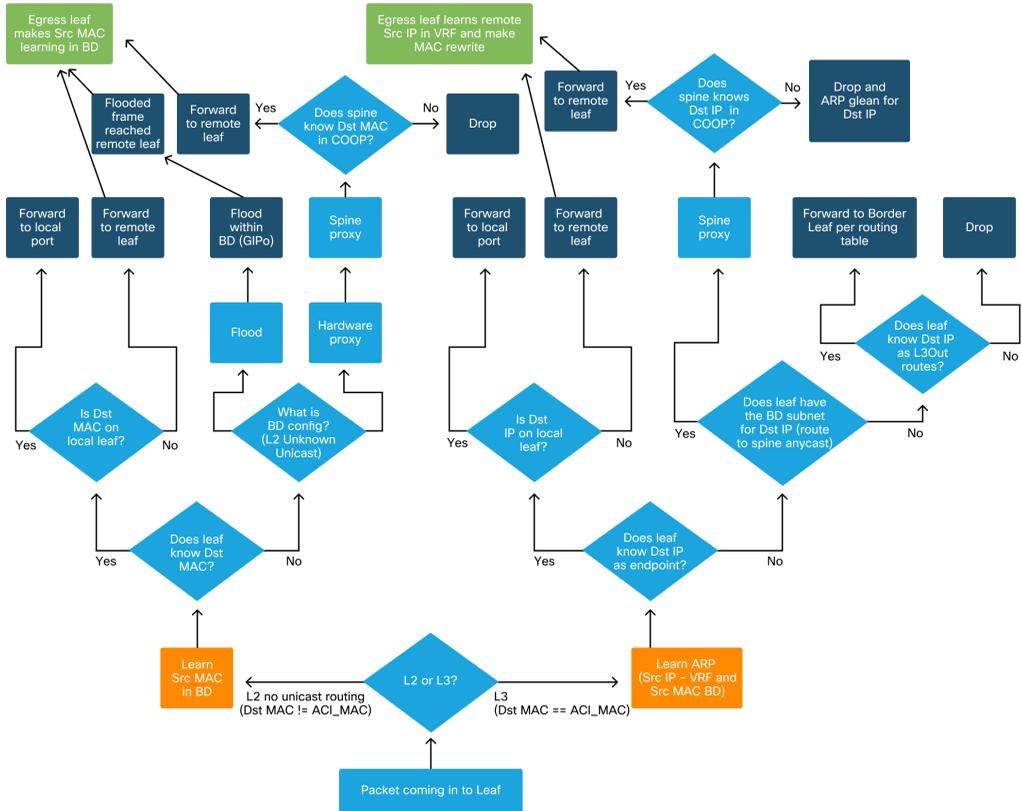
### COOP lookup on spine - destination IP is unknown

In the case where the COOP entry for the destination IP is not populated (silent endpoint or aged out), the spine will generate an ARP glean to resolve it. For more information, refer to "Multi-Pod Forwarding" section.

## Summary

The following drawing summarizes the ACI forwarding for Layer 2 and Layer 3 use case.

ACI forwarding summary

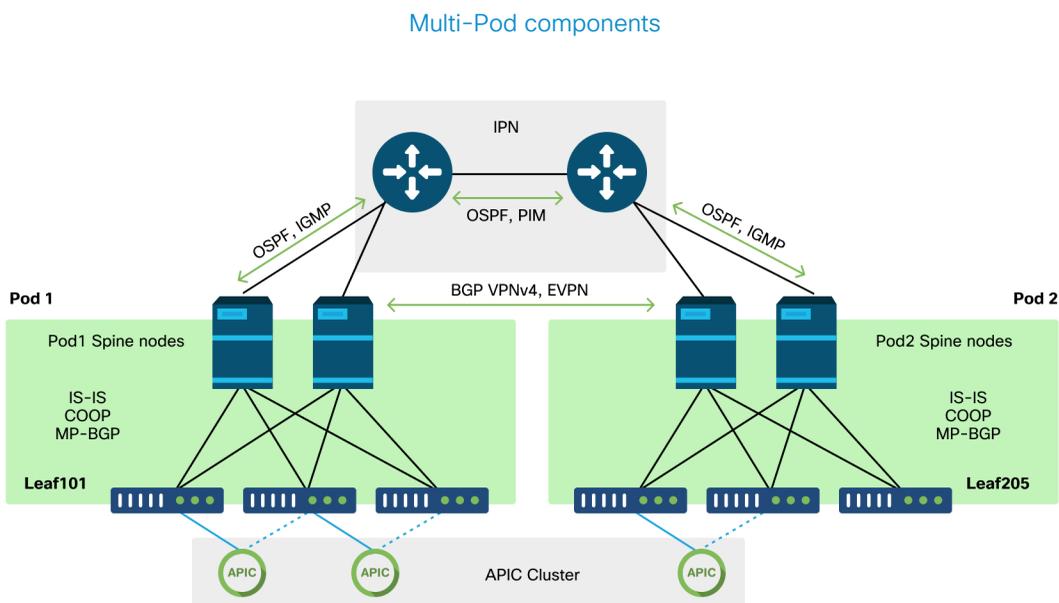


# Multi-Pod forwarding

## Multi-Pod forwarding overview

This chapter will cover how to troubleshoot scenarios in which connectivity is not working correctly across Pods in a Multi-Pod environment

Before looking at specific troubleshooting examples, it is important to take a moment to understand the Multi-Pod components at a high level.



Similar to a traditional ACI fabric, a Multi-Pod fabric is still considered to be a single ACI fabric and relies on a single APIC cluster for management.

Within each individual Pod, ACI leverages the same protocols in the overlay as a traditional fabric. This includes IS-IS for exchange of TEP information as well as

multicast Outgoing Interface (OIF) selection, COOP for a global endpoint repository, and BGP VPNv4 for the distribution of external routers through the fabric.

Multi-Pod builds on those components as it must connect each Pod together.

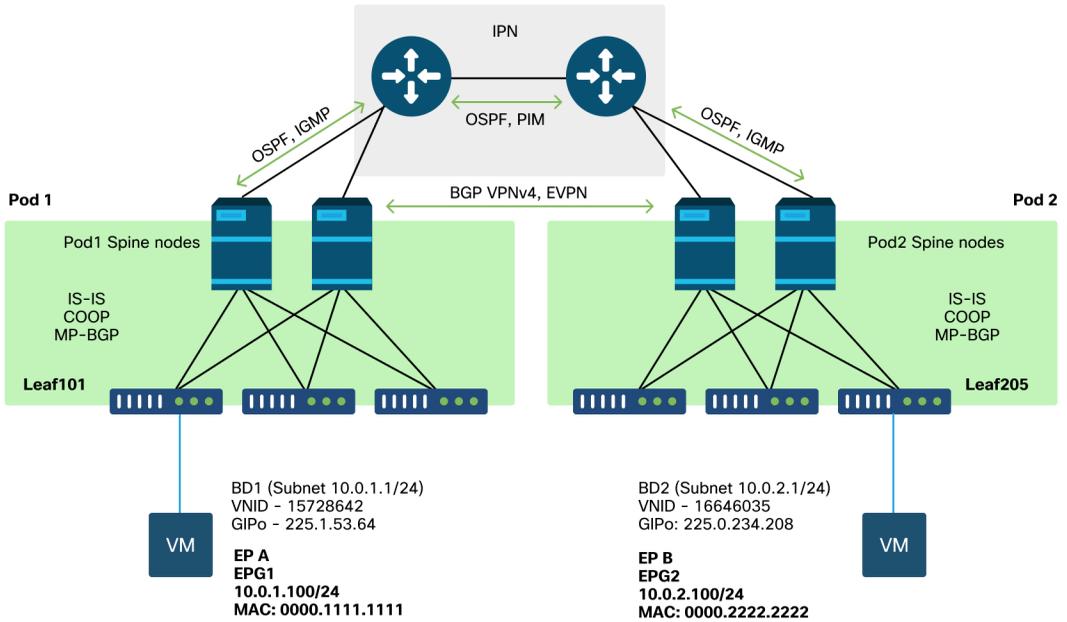
- To exchange routing information regarding TEPs in the remote Pod, OSPF is used to advertise the summary TEP pool through the IPN.
- To exchange external routes learned from one Pod to another, the BGP VPNv4 address-family is extended between spine nodes. Each Pod becomes a separate route-reflector cluster.
- To synchronize endpoints as well as other information stored in COOP across Pods, the BGP EVPN address-family is extended between spine nodes.
- Lastly, in order to handle the flooding of Broadcast, Unknown-Unicast, and Multicast (BUM) traffic across Pods, the spine nodes in each Pod act as IGMP hosts and the IPN routers exchange multicast routing information through Bidirectional PIM.

A large portion of the Multi-Pod troubleshooting scenarios and workflows are similar to Single Pod ACI fabrics. This Multi-Pod section will mostly focus on the differences between Single Pod and Multi-Pod forwarding.

## **Multi-Pod forwarding troubleshooting workflow**

As with troubleshooting any scenario, it is important to begin by understanding what the expected state is. The following topology will be used in the below examples.

Topology for Multi-Pod examples



The high level workflow is as follows:

- 1 Is the flow Unicast or multi-destination? Remember, even if the flow is expected to be unicast in the working state, if ARP isn't resolved then it is a multi-destination flow.
- 2 Is the flow routed or bridged? Traditionally, a routed flow from an ACI perspective would be any flow where the destination MAC address is the router MAC address that is owned by a gateway configured on ACI. Additionally, if ARP flooding is disabled, then the ingress leaf would route based on the target-IP address. If the destination MAC address is not owned by ACI, then the switch would either forward based on the MAC address or follow the 'unknown unicast' behavior configured on the bridge domain.
- 3 Is the ingress leaf dropping the flow? fTriage and ELAM are the best tools to confirm this.

If the flow is Layer 3 unicast:

- 1 Does the ingress leaf have an endpoint learn for the destination IP in the same VRF as the source EPG? If so, this will always take precedence over any learned routes. The leaf will forward directly to the tunnel address or egress interface where the endpoint is learned.
- 2 If there is no endpoint learn, does the ingress leaf have a route for the destination that has the 'Pervasive' flag set? This indicates that the destination subnet is configured as a Bridge Domain subnet and that the next-hop should be the spine proxy in the local Pod.
- 3 If there is no Pervasive route, then the last resort would be any routes that are learned through an L3Out. This portion is identical to Single Pod L3Out forwarding.

If the flow is Layer 2 unicast:

- 1 Does the ingress leaf have an endpoint learn for the destination MAC address in the same Bridge Domain as the source EPG? If so, the leaf will forward to the remote tunnel IP or out the local interface where the endpoint is learned.
- 2 If there is no learn for the destination MAC address in the source Bridge Domain, then the leaf will forward based on the 'unknown-unicast' behavior the BD is set to. If it is set to 'Flood', then the leaf will flood to the GIPo multicast group allocated for the Bridge Domain. Local and remote Pods should get a flooded copy. If it is set to 'Hardware Proxy' then the frame is sent to the spine for a proxy lookup and forwarded based on the spine's COOP entry.

Since the troubleshooting outputs would be considerably different for unicast compared to BUM, working outputs and scenarios for unicast will be considered before and then move to BUM.

## Multi-Pod unicast troubleshooting workflow

Following the topology, walk through the flow from 10.0.2.100 on leaf205 to 10.0.1.100 on leaf101.

Note, before proceeding here, it is important to confirm whether the source has ARP resolved for the gateway (for a routed flow) or the destination MAC address (for a bridged flow)

**1. Confirm that the ingress leaf receives the packet. Use the ELAM CLI tool shown in the "Tools" section along with the ereport output available in 4.2. The ELAM Assistant App is also used.**

```

module-1# debug platform internal tah elam asic 0
module-1(DBG-elam)# trigger reset
module-1(DBG-elam)# trigger init in-select 6 out-select 1
module-1(DBG-elam-insel6)# set outer ipv4 src_ip 10.0.2.100 dst_ip 10.0.1.100
module-1(DBG-elam-insel6)# start
module-1(DBG-elam-insel6)# stat
ELAM STATUS
=====
Asic 0 Slice 0 Status Armed
Asic 0 Slice 1 Status Triggered

```

Note that the ELAM triggered which confirms the packet was received on the ingress switch. Now look at a couple of fields in the report since the output is extensive.

```

=====
                                Captured Packet
=====

-----
Outer Packet Attributes
-----
Outer Packet Attributes      : l2uc ipv4 ip ipuc ipv4uc
Opcode                       : OPCODE_UC

-----
Outer L2 Header
-----
Destination MAC              : 0022.BDF8.19FF
Source MAC                   : 0000.2222.2222
802.1Q tag is valid         : yes( 0x1 )
CoS                           : 0( 0x0 )
Access Encap VLAN           : 1021( 0x3FD )

-----
Outer L3 Header
-----
L3 Type                      : IPv4
IP Version                   : 4
DSCP                         : 0
IP Packet Length             : 84 ( = IP header(28 bytes) + IP payload )
Don't Fragment Bit          : not set
TTL                          : 255
IP Protocol Number          : ICMP
IP CheckSum                  : 10988( 0x2AEC )
Destination IP               : 10.0.1.100
Source IP                    : 10.0.2.100

```

There is much more info in the ereport about where the packet is going but the ELAM Assistant App is currently more useful for interpreting this data. The ELAM Assistant output for this flow will be shown later in this chapter.

## 2. Is the ingress leaf learning the destination as an endpoint in the ingress VRF? If not, is there a route?

```
a-leaf205# show endpoint ip 10.0.1.100 detail
```

```
Legend:
```

```
s - arp          H - vtep          V - vpc-attached    p - peer-aged
R - peer-attached-r1 B - bounce      S - static          M - span
D - bounce-to-proxy O - peer-attached  a - local-aged     m - svc-mgr
L - local        E - shared-service
```

VLAN/ Domain	Encap VLAN	MAC Address IP Address	MAC Info/ IP Info	Interface	Endpoint Group Info

No output in the above command means the destination IP is not learned. Next check the routing table.

```
a-leaf205# show ip route 10.0.1.100 vrf Prod:Vrf1
```

```
IP Route Table for VRF "Prod:Vrf1"
```

```
'*' denotes best ucast next-hop
```

```
'**' denotes best mcast next-hop
```

```
'[x/y]' denotes [preference/metric]
```

```
'%<string>' in via output denotes VRF <string>
```

```
10.0.1.0/24, ubest/mbest: 1/0, attached, direct, pervasive
```

```
*via 10.0.120.34%overlay-1, [1/0], 01:55:37, static, tag 4294967294
```

```
recursive next hop: 10.0.120.34/32%overlay-1
```

In the above output, the pervasive flag is seen which indicates this is a Bridge Domain subnet route. The next-hop should be an anycast proxy address on the spines.

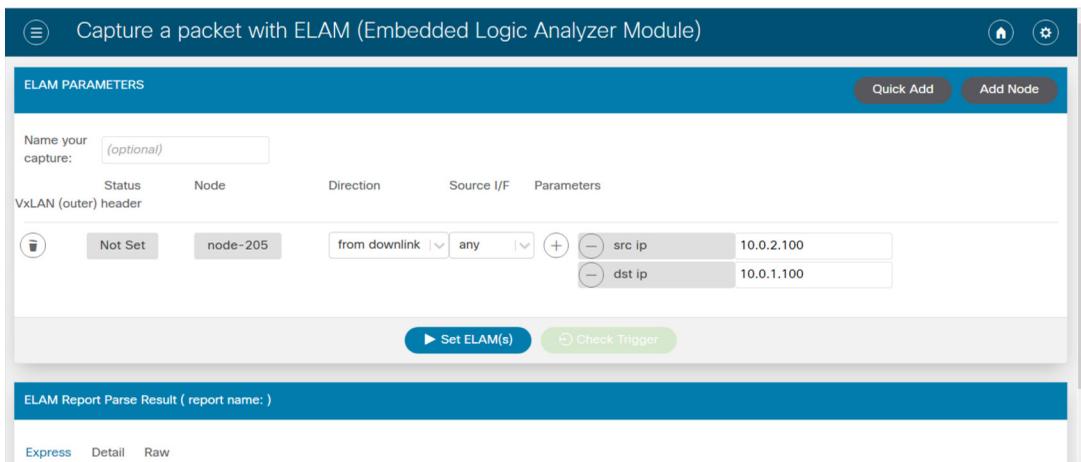
```
a-leaf205# show isis dtep vrf overlay-1 | grep 10.0.120.34
```

```
10.0.120.34      SPINE    N/A      PHYSICAL, PROXY-ACAST-V4
```

Note that if the endpoint is learned on a tunnel or physical interface, this will take precedence, causing the packet to be forwarded directly there. Refer to the "External forwarding" chapter of this book for more details.

Use the ELAM Assistant to confirm the forwarding decisions seen in the above outputs.

### ELAM Assistant configuration



### Verify forwarding decisions

Packet Forwarding Information	
<b>Forward Result</b>	
Destination Type	To another ACI node (LEAF, AVS/AVE etc.)
Destination TEP	10.0.120.34 (IPv4 Spine-Proxy)
Destination Physical Port	eth1/53
<b>Contract</b>	
Destination EPG pcTag (dclass)	0x1 / 1 (pcTag 1 is to ignore contract for special packets such as Spine-Proxy, ARP, Multicast etc.)
Source EPG pcTag (sclass)	0xC001 / 49153 (Prod:ap1-epg2)
Contract was applied	0 (Contract was not applied on this node)
<b>Drop</b>	
Drop Code	no drop

The above output shows that the ingress leaf is forwarding the packet to the IPv4 spine proxy address. This is what is expected to happen.

### 3. Confirm on the spine that the destination IP is present in COOP so that the proxy request works.

There are multiple ways to get the COOP output on the spine, for example, look at it with a 'show coop internal info ip-db' command:

```
a-spine4# show coop internal info ip-db | grep -B 2 -A 15 "10.0.1.100"
-----
IP address : 10.0.1.100
Vrf : 2392068 <--This vnid should correspond to vrf where the IP is learned. Check operational tab of the tenant
vrfs
Flags : 0x2
EP bd vnid : 15728642
EP mac : 00:00:11:11:11:11
Publisher Id : 192.168.1.254
Record timestamp : 12 31 1969 19:00:00 0
Publish timestamp : 12 31 1969 19:00:00 0
Seq No: 0
Remote publish timestamp: 09 30 2019 20:29:07 9900483
URIB Tunnel Info
Num tunnels : 1
    Tunnel address : 10.0.0.34 <--When learned from a remote pod this will be an External Proxy TEP. We'll
cover this more
    Tunnel ref count : 1
-----
```

Other commands to run on the spine:

```
Query COOP for 12 entry:
moquery -c coopEpRec -f 'coop.EpRec.mac=="00:00:11:11:22:22"
```

```
Query COOP for 13 entry and get parent 12 entry:
moquery -c coopEpRec -x rsp-subtree=children 'rsp-subtree-filter=eq(coopIpv4Rec.addr,"1.1.1.1")' rsp-subtree-include=required
```

```
Query COOP for 13 entry only:
moquery -c coopIpv4Rec -f 'coop.Ipv4Rec.addr=="1.1.1.1"'
```

The useful thing about the multiple moquery is that they can also be run directly on an APIC and the user can see every spine that has the record in coop.

#### 4. Multi-Pod spine proxy forwarding decision

If the spine's COOP entry points to a tunnel in the local Pod then forwarding is based on traditional ACI behavior.

Note that owner of a TEP can be verified in the fabric by running from an APIC:  
`moquery -c ipv4Addr -f 'ipv4.Addr.addr=="<tunnel address>"'`

In the above proxy scenario, the tunnel next-hop is 10.0.0.34. Who is the owner of this IP address?:

```
a-apic1# moquery -c ipv4Addr -f 'ipv4.Addr.addr=="10.0.0.34"' | grep dn
dn          : topology/pod-1/node-1002/sys/ipv4/inst/dom-overlay-1/if-[1o9]/addr-[10.0.0.34/32]
dn          : topology/pod-1/node-1001/sys/ipv4/inst/dom-overlay-1/if-[1o2]/addr-[10.0.0.34/32]
```

This IP is owned by both spine nodes in Pod 1. This is a specific IP called an External Proxy address. In the same way that ACI has proxy addresses owned by the spine nodes within a Pod (see step 2 of this section), there are also proxy addresses assigned to the Pod itself. This interface type can be verified by running:

```
a-apic1# moquery -c ipv4If -x rsp-subtree=children 'rsp-subtree-filter=eq(ipv4Addr.addr,"10.0.0.34")' rsp-
subtree-include=required

...
# ipv4.If
mode          : anycast-v4, external
# ipv4.Addr
addr          : 10.0.0.34/32
dn            : topology/pod-1/node-1002/sys/ipv4/inst/dom-overlay-1/if-[1o9]/addr-[10.0.0.34/32]
```

The 'external' flag indicates this is an external proxy TEP.

## 5. Verify BGP EVPN on the spine

The coop endpoint record should be imported from BGP EVPN on the spine. The following command can be used to verify that it is in EVPN (though if it is already in COOP with a next-hop of the remote Pod external proxy TEP it can be assumed it came from EVPN):

```
a-spine4# show bgp l2vpn evpn 10.0.1.100 vrf overlay-1
Route Distinguisher: 1:16777199
BGP routing table entry for [2]:[0]:[15728642]:[48]:[0000.1111.1111]:[32]:[10.0.1.100]/272, version 689242 dest
ptr 0xaf42a4ca
Paths: (2 available, best #2)
Flags: (0x000202 00000000) on xmit-list, is not in rib/evpn, is not in HW, is locked
Multipath: eBGP iBGP

  Path type: internal 0x40000018 0x2040 ref 0 adv path ref 0, path is valid, not best reason: Router Id, remote
  nh not installed
  AS-Path: NONE, path sourced internal to AS
  192.168.1.254 (metric 7) from 192.168.1.102 (192.168.1.102)
    Origin IGP, MED not set, localpref 100, weight 0
    Received label 15728642 2392068
    Received path-id 1
    Extcommunity:
      RT:5:16
      SOO:1:1
      ENCAP:8
      Router MAC:0200.0000.0000

    Advertised path-id 1
  Path type: internal 0x40000018 0x2040 ref 1 adv path ref 1, path is valid, is best path, remote nh not
  installed
  AS-Path: NONE, path sourced internal to AS
  192.168.1.254 (metric 7) from 192.168.1.101 (192.168.1.101)
    Origin IGP, MED not set, localpref 100, weight 0
    Received label 15728642 2392068
    Received path-id 1
    Extcommunity:
      RT:5:16
      SOO:1:1
      ENCAP:8
      Router MAC:0200.0000.0000

    Path-id 1 not advertised to any peer
```

Note that the above command can be run for a MAC address as well.

192.168.1.254 is the dataplane TEP configured during Multi-Pod setup. Note however that even though it is advertised in BGP as the NH, the actual next-hop will be the external proxy TEP.

192.168.1.101 and .102 are the Pod 1 spine nodes advertising this path.

## 6. Verify COOP on the spines in the destination Pod.

The same command as earlier can be used:

```
a-spine2# show coop internal info ip-db | grep -B 2 -A 15 "10.0.1.100"
```

```
-----
IP address : 10.0.1.100
Vrf : 2392068
Flags : 0
EP bd vnid : 15728642
EP mac : 00:50:56:81:3E:E6
Publisher Id : 10.0.72.67
Record timestamp : 10 01 2019 15:46:24 502206158
Publish timestamp : 10 01 2019 15:46:24 524378376
Seq No: 0
Remote publish timestamp: 12 31 1969 19:00:00 0
URIB Tunnel Info
Num tunnels : 1
    Tunnel address : 10.0.72.67
    Tunnel ref count : 1
-----
```

Verify who owns the tunnel address by running the following command on an APIC:

```
a-apic1# moquery -c ipv4Addr -f 'ipv4.Addr.addr=="10.0.72.67"'
Total Objects shown: 1

# ipv4.Addr
addr          : 10.0.72.67/32
childAction   :
ctrl         :
dn           : topology/pod-1/node-101/sys/ipv4/inst/dom-overlay-1/if-[lo0]/addr-[10.0.72.67/32]
ipv4CfgFailedBmp :
ipv4CfgFailedTs : 00:00:00:00.000
```

```

ipv4CfgState      : 0
lcOwn             : local
modTs             : 2019-09-30T18:42:43.262-04:00
monPolDn         : uni/fabric/monfab-default
operSt           : up
operStQual       : up
pref             : 0
rn               : addr-[10.0.72.67/32]
status           :
tag              : 0
type             : primary
vpcPeer         : 0.0.0.0

```

The above command shows that the tunnel from COOP points to leaf101. This means that leaf101 should have the local learn for the destination endpoint.

### 7. Verify that the egress leaf has the local learn.

This can be done via a 'show endpoint' command:

```

a-leaf101# show endpoint ip 10.0.1.100 detail
Legend:
s - arp          H - vtep          V - vpc-attached   p - peer-aged
R - peer-attached-r1 B - bounce      S - static         M - span
D - bounce-to-proxy 0 - peer-attached a - local-aged    m - svc-mgr
L - local        E - shared-service

```

VLAN/ Domain	Encap VLAN	MAC Address IP Address	MAC Info/ IP Info	Interface	Endpoint Group Info
341	vlan-1075	0000.1111.1111 LV		<b>po5</b>	Prod:ap1:epg1
Prod:Vrf1	vlan-1075	10.0.1.100 LV		<b>po5</b>	

Note that the endpoint is learned. The packet should be forwarded based out port-channel 5 with VLAN tag 1075 set.

## Using fTriage to verify the end-to-end flow

As discussed in the "Tools" section of this chapter, fTriage can be used to map out an existing flow end-to-end and understand what every switch in the path is doing with the packet. This is particularly useful in larger and more complex deployments such as Multi-Pod.

Note that fTriage will take some time to fully run (potentially 15 minutes).

When running fTriage on the example flow:

```
a-asic1# ftriage route -ii LEAF:205 -dip 10.0.1.100 -sip 10.0.2.100
fTriage Status: {"dbgFtriage": {"attributes": {"operState": "InProgress", "pid": "7297", "apicId": "1", "id": "0"}}}
Starting ftriage
Log file name for the current run is: ftlog.2019-10-01-16-04-15-438.txt
2019-10-01 16:04:15,442 INFO /controller/bin/ftriage route -ii LEAF:205 -dip 10.0.1.100 -sip 10.0.2.100
2019-10-01 16:04:38,883 INFO ftriage: main:1165 Invoking ftriage with default password and default username:
apic#fallback\admin
2019-10-01 16:04:54,678 INFO ftriage: main:839 L3 packet Seen on a-leaf205 Ingress: Eth1/31 Egress: Eth1/53 Vnid:
2392068
2019-10-01 16:04:54,896 INFO ftriage: main:242 ingress encap string vlan-1021
2019-10-01 16:04:54,899 INFO ftriage: main:271 Building ingress BD(s), Ctx
2019-10-01 16:04:56,778 INFO ftriage: main:294 Ingress BD(s) Prod:Bd2
2019-10-01 16:04:56,778 INFO ftriage: main:301 Ingress Ctx: Prod:Vrf1
2019-10-01 16:04:56,887 INFO ftriage: pktrec:490 a-leaf205: Collecting transient losses snapshot for LC module: 1
2019-10-01 16:05:22,458 INFO ftriage: main:933 SIP 10.0.2.100 DIP 10.0.1.100
2019-10-01 16:05:22,459 INFO ftriage: unicast:973 a-leaf205: <- is ingress node
2019-10-01 16:05:25,206 INFO ftriage: unicast:1215 a-leaf205: Dst EP is remote
2019-10-01 16:05:26,758 INFO ftriage: misc:657 a-leaf205: DMAC(00:22:BD:F8:19:FF) same as RMAC(00:22:BD:F8:19:FF)
2019-10-01 16:05:26,758 INFO ftriage: misc:659 a-leaf205: L3 packet getting routed/bounced in SUG
2019-10-01 16:05:27,030 INFO ftriage: misc:657 a-leaf205: Dst IP is present in SUG L3 tbl
2019-10-01 16:05:27,473 INFO ftriage: misc:657 a-leaf205: RxDMAC DIPO(10.0.72.67) is one of dst TEPs ['10.0.72.67']
2019-10-01 16:06:25,200 INFO ftriage: main:622 Found peer-node a-spine3 and IF: Eth1/31 in candidate list
2019-10-01 16:06:30,802 INFO ftriage: node:643 a-spine3: Extracted Internal-port GPD Info for lc: 1
2019-10-01 16:06:30,803 INFO ftriage: fcls:4414 a-spine3: LC trigger ELAM with IFS: Eth1/31 Asic :3 Slice: 1 Sroid: 24
2019-10-01 16:07:05,717 INFO ftriage: main:839 L3 packet Seen on a-spine3 Ingress: Eth1/31 Egress: LC-1/3 FC-24/0 Port-1
Vnid: 2392068
2019-10-01 16:07:05,718 INFO ftriage: pktrec:490 a-spine3: Collecting transient losses snapshot for LC module: 1
2019-10-01 16:07:28,043 INFO ftriage: fib:332 a-spine3: Transit in spine
2019-10-01 16:07:35,902 INFO ftriage: unicast:1252 a-spine3: Enter dbg_sub_nexthop with Transit inst: ig infra: False
glbs.dipo: 10.0.72.67
2019-10-01 16:07:36,018 INFO ftriage: unicast:1417 a-spine3: EP is known in COOP (DIPO = 10.0.72.67)
2019-10-01 16:07:40,422 INFO ftriage: unicast:1458 a-spine3: Infra route 10.0.72.67 present in RIB
2019-10-01 16:07:40,423 INFO ftriage: node:1331 a-spine3: Mapped LC interface: LC-1/3 FC-24/0 Port-1 to FC interface: FC-
```

```

24/0 LC-1/3 Port-1
2019-10-01 16:07:46,059 INFO ftriage: node:460 a-spine3: Extracted GPD Info for fc: 24
2019-10-01 16:07:46,060 INFO ftriage: fcls:5748 a-spine3: FC trigger ELAM with IFS: FC-24/0 LC-1/3 Port-1 Asic :0 Slice: 1
Srcid: 40
2019-10-01 16:08:06,735 INFO ftriage: unicast:1774 L3 packet Seen on FC of node: a-spine3 with Ingress: FC-24/0 LC-1/3 Port-
1 Egress: FC-24/0 LC-1/3 Port-1 Vnid: 2392068
2019-10-01 16:08:06,735 INFO ftriage: pktrec:487 a-spine3: Collecting transient losses snapshot for FC module: 24
2019-10-01 16:08:09,123 INFO ftriage: node:1339 a-spine3: Mapped FC interface: FC-24/0 LC-1/3 Port-1 to LC interface: LC-
1/3 FC-24/0 Port-1
2019-10-01 16:08:09,124 INFO ftriage: unicast:1474 a-spine3: Capturing Spine Transit pkt-type L3 packet on egress LC on
Node: a-spine3 IFS: LC-1/3 FC-24/0 Port-1
2019-10-01 16:08:09,594 INFO ftriage: fcls:4414 a-spine3: LC trigger ELAM with IFS: LC-1/3 FC-24/0 Port-1 Asic :3 Slice: 1
Srcid: 48
2019-10-01 16:08:44,447 INFO ftriage: unicast:1510 a-spine3: L3 packet Spine egress Transit pkt Seen on a-spine3 Ingress:
LC-1/3 FC-24/0 Port-1 Egress: Eth1/29 Vnid: 2392068
2019-10-01 16:08:44,448 INFO ftriage: pktrec:490 a-spine3: Collecting transient losses snapshot for LC module: 1
2019-10-01 16:08:46,691 INFO ftriage: unicast:1681 a-spine3: Packet is exiting the fabric through {a-spine3: ['Eth1/29']}
Dipo 10.0.72.67 and filter SIP 10.0.2.100 DIP 10.0.1.100
2019-10-01 16:10:19,947 INFO ftriage: main:716 Capturing L3 packet Fex: False on node: a-spine1 IF: Eth2/25
2019-10-01 16:10:25,752 INFO ftriage: node:643 a-spine1: Extracted Internal-port GPD Info for lc: 2
2019-10-01 16:10:25,754 INFO ftriage: fcls:4414 a-spine1: LC trigger ELAM with IFS: Eth2/25 Asic :3 Slice: 0 Srcid: 24
2019-10-01 16:10:51,164 INFO ftriage: main:716 Capturing L3 packet Fex: False on node: a-spine2 IF: Eth1/31
2019-10-01 16:11:09,690 INFO ftriage: main:839 L3 packet Seen on a-spine2 Ingress: Eth1/31 Egress: Eth1/25 Vnid: 2392068
2019-10-01 16:11:09,690 INFO ftriage: pktrec:490 a-spine2: Collecting transient losses snapshot for LC module: 1
2019-10-01 16:11:24,882 INFO ftriage: fib:332 a-spine2: Transit in spine
2019-10-01 16:11:32,598 INFO ftriage: unicast:1252 a-spine2: Enter dbg_sub_nexthop with Transit inst: ig infra: False
glbs.dipo: 10.0.72.67
2019-10-01 16:11:32,714 INFO ftriage: unicast:1417 a-spine2: EP is known in COOP (DIPO = 10.0.72.67)
2019-10-01 16:11:36,901 INFO ftriage: unicast:1458 a-spine2: Infra route 10.0.72.67 present in RIB
2019-10-01 16:11:47,106 INFO ftriage: main:622 Found peer-node a-leaf101 and IF: Eth1/54 in candidate list
2019-10-01 16:12:09,836 INFO ftriage: main:839 L3 packet Seen on a-leaf101 Ingress: Eth1/54 Egress: Eth1/30 (Po5) Vnid:
11470
2019-10-01 16:12:09,952 INFO ftriage: pktrec:490 a-leaf101: Collecting transient losses snapshot for LC module: 1
2019-10-01 16:12:30,991 INFO ftriage: nxos:1404 a-leaf101: nxos matching rule id:4659 scope:84 filter:65534
2019-10-01 16:12:32,327 INFO ftriage: main:522 Computed egress encap string vlan-1075
2019-10-01 16:12:32,333 INFO ftriage: main:313 Building egress BD(s), Ctx
2019-10-01 16:12:34,559 INFO ftriage: main:331 Egress Ctx Prod:Vrf1
2019-10-01 16:12:34,560 INFO ftriage: main:332 Egress BD(s): Prod:Bd1
2019-10-01 16:12:37,704 INFO ftriage: unicast:1252 a-leaf101: Enter dbg_sub_nexthop with Local inst: eg infra: False
glbs.dipo: 10.0.72.67
2019-10-01 16:12:37,705 INFO ftriage: unicast:1257 a-leaf101: dbg_sub_nexthop invokes dbg_sub_eg for ptep
2019-10-01 16:12:37,705 INFO ftriage: unicast:1784 a-leaf101: <- is egress node
2019-10-01 16:12:37,911 INFO ftriage: unicast:1833 a-leaf101: Dst EP is local
2019-10-01 16:12:37,912 INFO ftriage: misc:657 a-leaf101: EP if(Po5) same as egr if(Po5)
2019-10-01 16:12:38,172 INFO ftriage: misc:657 a-leaf101: Dst IP is present in SUG L3 tbl
2019-10-01 16:12:38,564 INFO ftriage: misc:657 a-leaf101: RW seg_id:11470 in SUG same as EP segid:11470
fTriage Status: {"dbgFtriage": {"attributes": {"operState": "Idle", "pid": "0", "apicId": "0", "id": "0"}}}
fTriage Status: {"dbgFtriage": {"attributes": {"operState": "Idle", "pid": "0", "apicId": "0", "id": "0"}}}

```

There is a large amount of data in the fTriage. Some of the most important fields are highlighted. Note that the path of the packet was 'leaf205 (Pod 2) > spine3 (Pod 2) > spine2 (Pod 1) > leaf101 (Pod 1)'. All forwarding decisions and contract lookups made along the way are also visible.

Note that if this was a Layer 2 flow, the syntax of the fTriage would need to be set to something like:

```
ftrriage bridge -ii LEAF:205 -dmac 00:00:11:11:22:22
```

### Proxied requests where the EP is not in COOP

Before considering specific failure scenarios, there is one more piece to discuss related to unicast forwarding over Multi-Pod. What happens if the destination endpoint is unknown, the request is proxied, and the endpoint is not in COOP?

In this scenario, the packet/frame is sent to the spine and a glean request is generated.

When the spine generates a glean request, the original packet is still preserved in the request however, the packet receives ethertype 0xfff2 which is a Custom Ethertype reserved for gleans. For this reason, it will not be easy to interpret these messages in packet capture tools such as Wireshark.

The outer Layer 3 destination is also set to 239.255.255.240 which is a reserved multicast group specifically for glean messages. These should be flooded across the fabric and any egress leaf switches that have the destination subnet of the glean request deployed will generate an ARP request to resolve the destination. These ARPs are sent from the BD Subnet IP Address configured (therefore proxy requests can't resolve the location of Silent/Unknown endpoints if Unicast Routing is disabled on a Bridge Domain).

The reception of the glean message on the egress leaf and the subsequently generated ARP and received ARP response can be verified through the following command:

## Glean ARP verification

```

a-leaf205# show ip arp internal event-history event | grep -F -B 1 192.168.21.11
73) Event:E_DEBUG_DSF, length:127, at 316928 usecs after Wed May 1 08:31:53 2019
Updating epm ifidx: 1a01e000 vlan: 105 ip: 192.168.21.11, ifMode: 128 mac: 8c60.4f02.88fc <<< Endpoint is
learned
75) Event:E_DEBUG_DSF, length:152, at 316420 usecs after Wed May 1 08:31:53 2019
log_collect_arp_pkt; sip = 192.168.21.11; dip = 192.168.21.254; interface = Vlan104;info = Garp Check adj:(nil)
<<< Response received
77) Event:E_DEBUG_DSF, length:142, at 131918 usecs after Wed May 1 08:28:36 2019
log_collect_arp_pkt; dip = 192.168.21.11; interface = Vlan104;iod = 138; Info = Internal Request Done <<< ARP
request is generated by leaf
78) Event:E_DEBUG_DSF, length:136, at 131757 usecs after Wed May 1 08:28:36 2019 <<< Glean received, Dst IP is
in BD subnet
log_collect_arp_glean;dip = 192.168.21.11;interface = Vlan104;info = Received pkt Fabric-Glean: 1
79) Event:E_DEBUG_DSF, length:174, at 131748 usecs after Wed May 1 08:28:36 2019
log_collect_arp_glean; dip = 192.168.21.11; interface = Vlan104; vrf = CiscoLive2019:vrf1; info = Address in
PSVI subnet or special VIP <<< Glean Received, Dst IP is in BD subnet

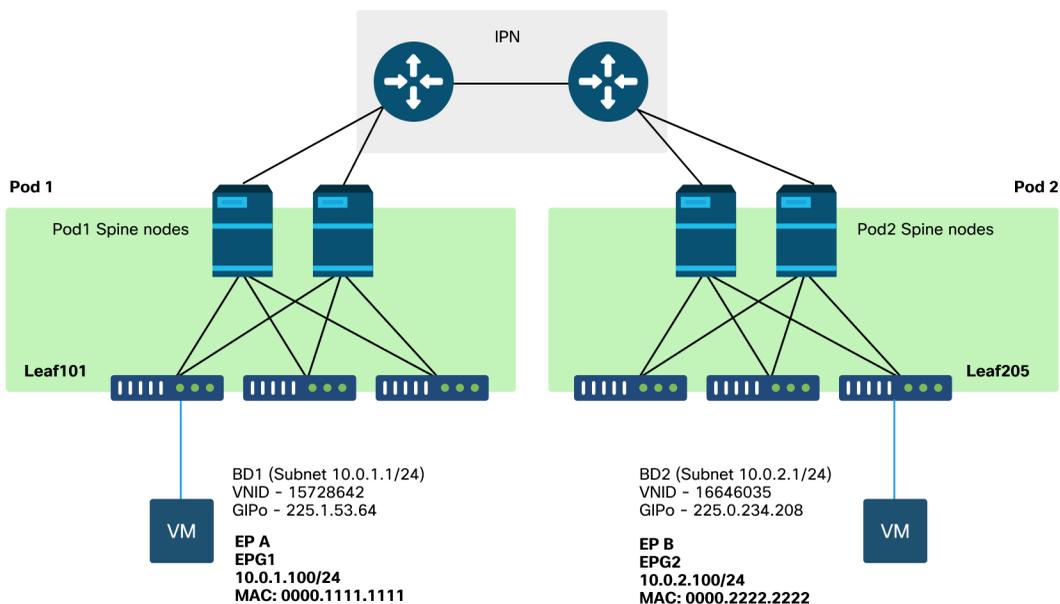
```

For reference, glean messages being sent to 239.255.255.240 is why this group needs to be included in the Bidirectional PIM group range on the IPN.

## Multi-Pod troubleshooting scenario #1

In the following topology, EP B cannot communicate with EP A.

Troubleshooting topology



Note that many of the problems seen for Multi-Pod forwarding are identical to problems seen in a Single Pod. For this reason, problems specific to Multi-Pod are focused on.

While following the unicast troubleshooting workflow described earlier, note that the request is proxied but the spine nodes in Pod 2 do not have the destination IP in COOP.

**Cause:**

As discussed earlier, COOP entries for remote Pod endpoints are populated from BGP EVPN information. As a result, it is important to determine:

a.) Does the source Pod (Pod 2) spine have it in EVPN?

```
a-spine4# show bgp 12vpn evpn 10.0.1.100 vrf overlay-1
<no output>
```

b.) Does the remote Pod (Pod 1) spine have it in EVPN?

```
a-spine1# show bgp 12vpn evpn 10.0.1.100 vrf overlay-1
Route Distinguisher: 1:16777199 (L2VNI 1)
BGP routing table entry for [2]:[0]:[15728642]:[48]:[0050.5681.3ee6]:[32]:[10.0.1.100]/272, version 11751 dest
ptr 0xafbf8192
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 adv path ref 1, path is valid, is best path
AS-Path: NONE, path locally originated
  0.0.0.0 (metric 0) from 0.0.0.0 (192.168.1.101)
    Origin IGP, MED not set, localpref 100, weight 32768
    Received label 15728642 2392068
    Extcommunity:
      RT:5:16

    Path-id 1 advertised to peers:
```

The Pod 1 spine has it and the next-hop IP is 0.0.0.0; this means it was exported from COOP locally. Note, however, that the 'Advertised to peers' section does not include the Pod 2 spine nodes.

c.) Is BGP EVPN up between Pods?

```
a-spine4# show bgp l2vpn evpn summ vrf overlay-1
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
192.168.1.101	4	65000	57380	66362		0	0	0 00:00:21	<b>Active</b>
192.168.1.102	4	65000	57568	66357		0	0	0 00:00:22	<b>Active</b>

Notice in the above output that the BGP EVPN peerings are down between Pods. Anything besides a numeric value in the State/PfxRcd column indicates that the adjacency is not up. Pod 1 EPs aren't learned through EVPN and aren't imported into COOP.

If this issue is seen verify the following:

- 1 Is OSPF up between the spine nodes and the connected IPNs?
- 2 Do the spine nodes have routes learned through OSPF for the remote spine IPs?
- 3 Does the full path across the IPN support jumbo MTU?
- 4 Are all protocol adjacencies stable?

#### Other possible causes

If the endpoint is not in the COOP database of any Pod and the destination device is a silent host (not learned on any leaf switch in the fabric), verify that the fabric glean process is working correctly. For this to work:

- Unicast Routing must be enabled on the BD.
- The destination must be in a BD subnet.
- The IPN must be providing multicast routing service for the 239.255.255.240 group.

The multicast portion is covered more in the next section.

## **Multi-Pod broadcast, unknown unicast, and multicast (BUM) forwarding overview**

In ACI, traffic is flooded via overlay multicast groups in many different scenarios. For example, flooding occurs for:

- Multicast and broadcast traffic.
- Unknown unicast that must be flooded.
- Fabric ARP glean messages.
- EP announce messages.

Many features and functionality rely on BUM forwarding.

Within ACI, all Bridge Domains are allocated a multicast address known as a Group IP Outer (or GIPO) address. All traffic that must be flooded within a Bridge Domain is flooded on this GIPO.

## BD GIPo in GUI

The screenshot shows the Cisco APIC (CALO-A) interface. The left sidebar displays a navigation tree with 'Prod' selected, and 'Networking' > 'Bridge Domains' highlighted. The main content area, titled 'Networking - Bridge Domains', contains a table with the following data:

Name	Alias	Type	Segment	VRF	Multicast Address	Custom MAC Address
Bd1		regular	15728642	Vrf1	225.1.53.64	00:22:BD:F8:19:FF
Bd2		regular	16646035	Vrf1	225.0.234.208	00:22:BD:F8:19:FF

At the bottom of the table, there is a pagination control showing 'Page 1 of 1' and 'Objects Per Page: 15'.

The object can be queried directly on one of the APICs.

## BD GIPo in Moquery

```
a-apic1# moquery -c fvBD -f 'fv.BD.name=="Bd1"'
Total Objects shown: 1

# fv.BD
name           : Bd1
OptimizeWanBandwidth : no
annotation     :
arpFlood       : yes
bcastP       : 225.1.53.64
childAction    :
configIssues   :
descr          :
```

```

dn                : uni/tn-Prod/BD-Bd1
epClear           : no
epMoveDetectMode  :
epMoveDetectMode :
extMngdBy         :
hostBasedRouting  : no
intersiteBumTrafficAllow : no
intersiteL2Stretch : no
iplearning        : yes
ipv6McastAllow    : no
lcOwn             : local
limitIpLearnToSubnets : yes
llAddr           : ::
mac               : 00:22:BD:F8:19:FF
mcastAllow        : no
modTs             : 2019-09-30T20:12:01.339-04:00
monPolDn          : uni/tn-common/monepg-default
mtu               : inherit
multiDstPktAct    : bd-flood
nameAlias         :
ownerKey          :
ownerTag          :
pcTag             : 16387
rn                : BD-Bd1
scope             : 2392068
seg               : 15728642
status            :
type              : regular
uid               : 16011
unicastRoute      : yes
unkMacUcastAct   : proxy
unkMcastAct       : flood
v6unkMcastAct     : flood
vmac              : not-applicable

```

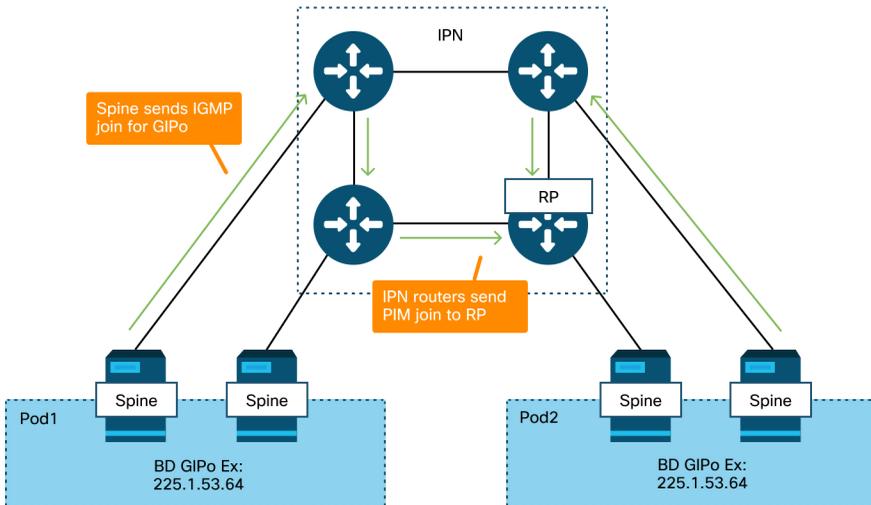
The above information about GIPo flooding is true regardless whether Multi-Pod is used or not. The additional portion of this that pertains to Multi-Pod is the multicast routing on the IPN.

IPN Multicast Routing involves the following:

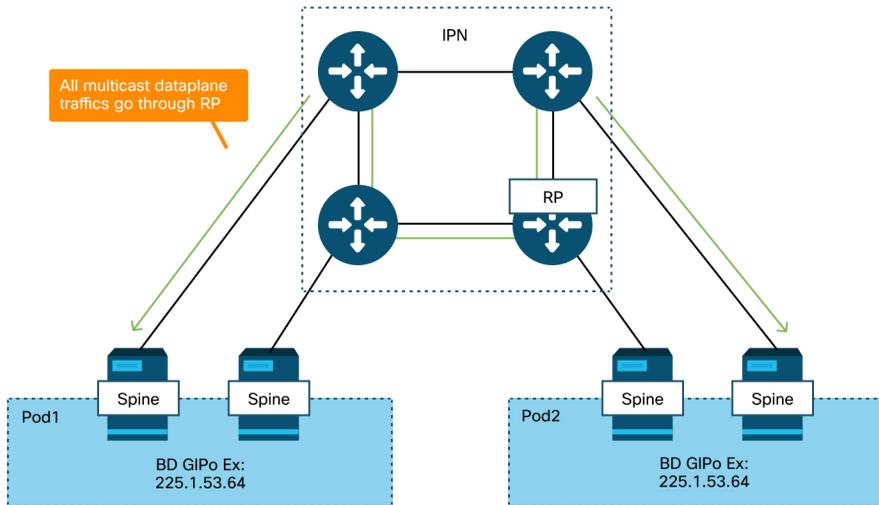
- Spine nodes act as multicast hosts (IGMP only). They do not run PIM.
- If a BD is deployed in a Pod, then one spine from that pod will send an IGMP join on one of its IPN-facing interfaces. This functionality is striped across all spine nodes and IPN-facing interface over many groups.

- The IPNs receive these joins and send PIM joins towards the Bidirectional PIM RP.
- Because PIM Bidir is used, there are no (S,G) trees. Only (\*,G) trees are used in PIM Bidir.
- All dataplane traffic sent to the GIPO goes through the RP.

IPN multicast control plane



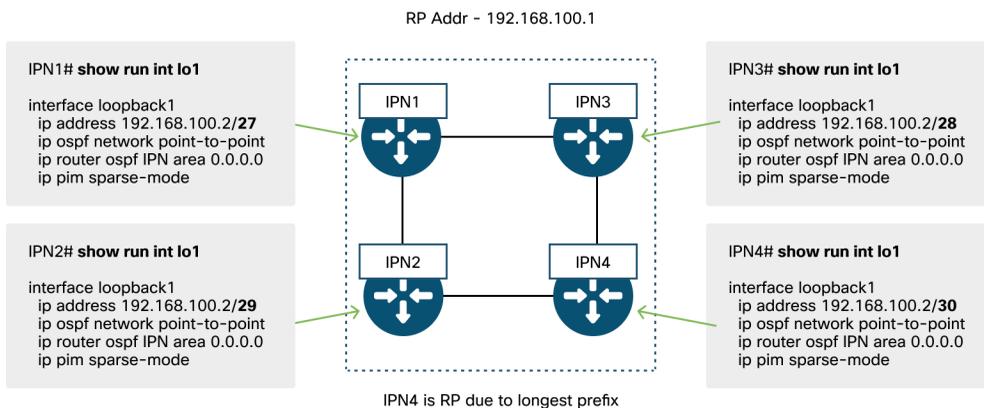
## IPN multicast dataplane



The only means of RP redundancy with PIM Bidir is to use Phantom. This is covered in detail within the Multi-Pod Discovery portion of this book. As a quick summary, note that with Phantom RP:

- All IPNs must be configured with the same RP address.
- The exact RP address must not exist on any device.
- Multiple devices advertise reachability to the subnet that contains the Phantom RP IP address. The advertised subnets should vary in subnet length so that all routers agree on who is advertising the best path for the RP. If this path is lost then convergence is dependent on the IGP.

## Phantom RP configuration



## Multi-Pod broadcast, unknown unicast, and multicast (BUM) troubleshooting workflow

- First confirm if the flow is truly being treated as multi-destination by the fabric. The flow will be flooded in the BD in these common examples:
  - The frame is an ARP broadcast and ARP flooding is enabled on the BD.
  - The frame is destined to a multicast group. Note that even if IGMP-snooping is enabled, the traffic is still always flooded into the fabric on the GIPo.
  - The traffic is destined to a multicast group that ACI is providing multicast routing services for.
  - The flow is a Layer 2 (bridged flow) and the destination MAC address is unknown and the unknown unicast behavior on the BD is set to 'Flood'.

The easiest way to determine which forwarding decision will be made is with an ELAM.

### ELAM ARP on ingress leaf

Packet Forwarding Information	
<b>Forward Result</b>	
Destination Type	To another ACI node (LEAF, AVS/AVE etc.)
Destination TEP	10.0.120.34 (IPv4 Spine-Proxy)
Destination Physical Port	eth1/53
<b>Contract</b>	
Destination EPG pcTag (dclass)	0x1 / 1 (pcTag 1 is to ignore contract for special packets such as Spine-Proxy, ARP, Multicast etc..)
Source EPG pcTag (sclass)	0xC001 / 49153 (Prod.ap1:epg2)
Contract was applied	0 (Contract was not applied on this node)
<b>Drop</b>	
Drop Code	no drop

- Once it is identified that the packet should be flooded in the BD, identify the BD GIPo. Refer to the section earlier in this chapter that talks about this. Spine ELAMs can also be run through the ELAM Assistant App to verify that the flooded traffic is being received.
- Once the GIPo is known, the last remaining portion (if there is no flooded traffic on the destination) is to verify the multicast routing tables on the IPN for that GIPo. The outputs to do this would vary depending on the IPN platform in use, but at a high level:
  - All IPN routers must agree on the RP and the RPF for this GIPo must point to this tree.
  - One IPN router connected to each Pod should be getting an IGMP join for the group.

### Multi-Pod troubleshooting scenario #2

This scenario would cover any scenario that involves ARP not being resolved across Multi-Pod or BUM scenarios (unknown unicast, etc.).

There are several common possible causes here.

#### **Possible cause 1: Multiple routers own the PIM RP address**

With this scenario, the ingress leaf floods the traffic (verify with ELAM), the source Pod receives and floods the traffic, but the remote Pod does not get it. For some BDs, flooding works, but for others it doesn't.

On the IPN, run 'show ip mroute <GIPO address>' for the GIPO to see that the RPF tree points to multiple, different routers.

If this is the case check the following:

- Verify that the actual PIM RP address isn't configured anywhere. Any device that owns that actual RP address would see a local /32 route for it.
- Verify that multiple IPN routers aren't advertising the same prefix length for the RP in the Phantom RP scenario.

#### **Possible cause 2: IPN routers aren't learning routes for the RP Address**

In the same way as the first possible cause, here the flooded traffic is failing to leave the IPN. The output of 'show ip route <rp address>' on each IPN router would show the locally configured prefix-length only rather than what the other routers are advertising.

The result of this is that each device thinks it is the RP even though the real RP IP address isn't configured anywhere.

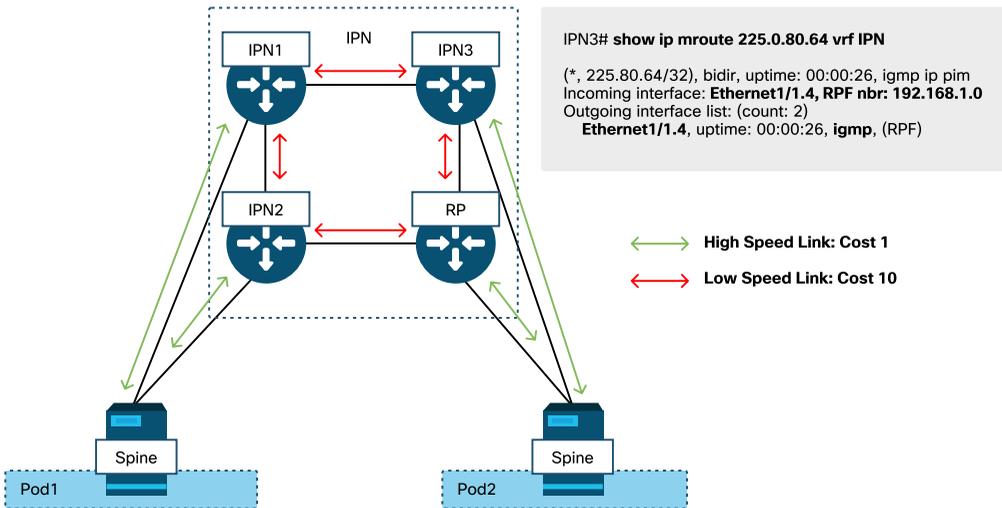
If this is the case. check the following:

- Verify that routing adjacencies are up between IPN routers. Verify that the route is in the actual protocol database (such as the OSPF database).

- Verify that all loopbacks that are supposed to be candidate RP's are configured as OSPF point-to-point network types. If this network type is not configured then each router will always advertise a /32 prefix-length regardless of what is actually configured.

**Possible cause 3: IPN routers aren't installing the GIPo route or the RPF points to ACI**  
 As mentioned earlier, ACI does not run PIM on its IPN-facing links. This means that the IPN's best path towards the RP should never point to ACI. The scenario where this could happen would be if multiple IPN routers are connected to the same spine and a better OSPF metric is seen through the spine than directly between IPN routers.

RPF interface toward ACI



To resolve this issue:

- Ensure that routing protocol adjacencies between IPN routers are up.
- Increase the OSPF cost metrics for the IPN-facing links on the spine nodes to a value that will make that metric less preferable than the IPN-to-IPN links.

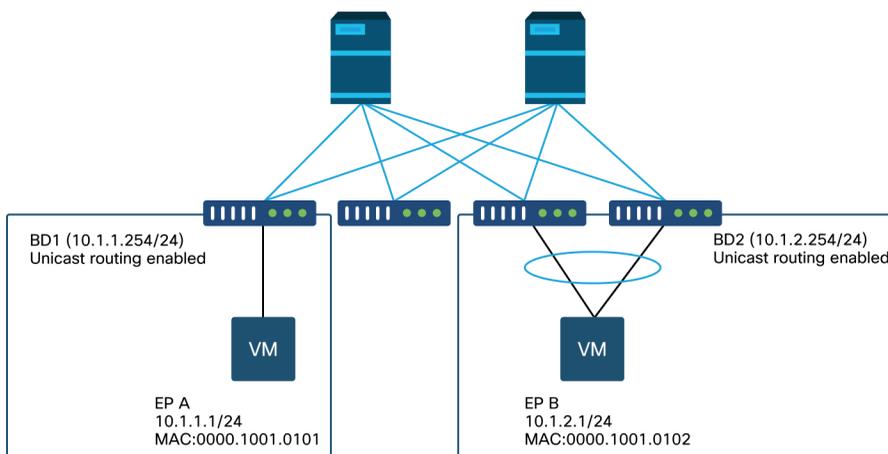
## Other references

Prior to ACI software 4.0, some challenges were experienced regarding the usage of COS 6 by external devices. Most of these issues have been addressed through 4.0 enhancements but for more info, please refer to CiscoLive session "BRKACI-2934 - Troubleshooting Multi-Pod" and the "Quality of Service" section.

# Intermittent drops

This chapter explains a troubleshooting example for an intermittent traffic drop.

Topology example



In this example, ping from EP A (10.1.1.1) to EP B (10.1.2.1) is experiencing the intermittent drops.

```
[EP-A ~]$ ping 10.1.2.1 -c 10
PING 10.1.2.1 (10.1.2.1) 56(84) bytes of data.
64 bytes from 10.1.2.1: icmp_seq=1 ttl=231 time=142 ms
64 bytes from 10.1.2.1: icmp_seq=2 ttl=231 time=141 ms
    <-- missing icmp_seq=3

64 bytes from 10.1.2.1: icmp_seq=4 ttl=231 time=141 ms
64 bytes from 10.1.2.1: icmp_seq=5 ttl=231 time=141 ms
64 bytes from 10.1.2.1: icmp_seq=6 ttl=231 time=141 ms
    <-- missing icmp_seq=7

64 bytes from 10.1.2.1: icmp_seq=8 ttl=231 time=176 ms
```

```
64 bytes from 10.1.2.1: icmp_seq=9 ttl=231 time=141 ms
64 bytes from 10.1.2.1: icmp_seq=10 ttl=231 time=141 ms

--- 10.1.2.1 ping statistics ---
10 packets transmitted, 8 received, 20% packet loss, time 9012ms
```

## Troubleshooting workflow

### 1. Determine which direction is causing the intermittent drops

Perform a packet capture (tcpdump, Wireshark, etc.) on the destination host (EP B). For ICMP, focus on the sequence number to see the intermittently dropped packets are observed on EP B.

```
[admin@EP-B ~]$ tcpdump -ni eth0 icmp
11:32:26.540957 IP 10.1.1.1 > 10.1.2.1: ICMP echo request, id 3569, seq 1, length 64
11:32:26.681981 IP 10.1.2.1 > 10.1.1.1: ICMP echo reply, id 3569, seq 1, length 64
11:32:27.542175 IP 10.1.1.1 > 10.1.2.1: ICMP echo request, id 3569, seq 2, length 64
11:32:27.683078 IP 10.1.2.1 > 10.1.1.1: ICMP echo reply, id 3569, seq 2, length 64
11:32:28.543173 IP 10.1.1.1 > 10.1.2.1: ICMP echo request, id 3569, seq 3, length 64 <---
11:32:28.683851 IP 10.1.2.1 > 10.1.1.1: ICMP echo reply, id 3569, seq 3, length 64 <---
11:32:29.544931 IP 10.1.1.1 > 10.1.2.1: ICMP echo request, id 3569, seq 4, length 64
11:32:29.685783 IP 10.1.2.1 > 10.1.1.1: ICMP echo reply, id 3569, seq 4, length 64
11:32:30.546860 IP 10.1.1.1 > 10.1.2.1: ICMP echo request, id 3569, seq 5, length 64
...
```

- Pattern 1 - All packets are observed on EP B packet capture.

Drops should be in ICMP echo reply (EP B to EP A).

- Pattern 2 - The intermittent drops are observed on EP B packet capture.

Drops should be in ICMP echo (EP A to EP B).

### 2. Check if another protocol with the same source/destination IP has the same issue

If possible, try to test the connectivity between the two endpoints using a different protocol allowed by the contract between them (such as ssh, telnet, http,..)

- Pattern 1 - Other protocols have the same intermittent drop.

The issue could be in endpoint flapping or queuing/buffering as shown below.

- Pattern 2 - Only ICMP has the intermittent drop.

The forwarding tables (such as endpoint table) should have no issue since forwarding is based on MAC and IP. Queuing/buffering should not be the reason either, as this would affect other protocols. The only reason that ACI would make a different forwarding decision based on protocol would be the PBR use-case.

One possibility is that one of spine nodes may have an issue. When a protocol is different, the packet with same source and destination could be load balanced to another uplink/fabric port (i.e. another spine) by the ingress leaf.

Atomic Counters can be used to ensure packets are not dropped on spine nodes and reach to the egress leaf. In case the packets didn't reach the egress leaf, check the ELAM on the ingress leaf to see which fabric port the packets are sent out. To isolate the issue to a specific spine, leaf uplinks can be shut down to force the traffic towards another spine.

### 3. Check if it's related to an endpoint learning issue

ACI uses an endpoint table to forward packets from one endpoint to another endpoint. An intermittent reachability issue can be caused by endpoint flapping because inappropriate endpoint information will cause the packet to be sent out to a wrong destination or to be classified into a wrong EPG that may result in a contract drop. Even if the destination is supposed to be an L3Out instead of an endpoint group, ensure that the IP is not learned as an endpoint in the same VRF across any leaf switches.

See the "Endpoint Flapping" sub-section in this section for more details on how to troubleshoot endpoint flapping.

### 4. Check if it's related to buffering issues by changing the traffic frequency

Increase or decrease the interval of ping to see if the drop ratio changes. The interval difference should be large enough.

In Linux, '-i' option can be used to change the interval (sec):

```
[EP-A ~]$ ping 10.1.2.1 -c 10 -i 5 -- Increase it to 5 sec  
[EP-A ~]$ ping 10.1.2.1 -c 10 -i 0.2 -- Decrease it to 0.2 msec
```

If the drop ratio increases when the interval is decreased, it is likely related to queuing or buffering on endpoints or switches.

The drop ratio to consider is (number of drops/total packets sent) instead of the (number of drops/time).

In such scenario, check the following.

- 1 Check if any drop counters on switch interfaces are increasing along with the ping. See "Interface drops" section in the chapter "Intra-Fabric forwarding" for details.
- 2 Check if the Rx counter is increasing along with the packets on the destination endpoint. If the Rx counter is increased with the same number as the transmitted packets, packets are likely being dropped on the endpoint itself. This could be due to endpoint buffering on TCP/IP stack.

For example, if 100000 pings are sent with as short interval as possible, the Rx counter on the endpoint can be observed as it increments by 100000.

```
[EP-B ~]$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.1.2.1 netmask 255.255.255.0 broadcast 10.1.2.255
ether 00:00:10:01:01:02 txqueuelen 1000 (Ethernet)
RX packets 101105 bytes 1829041
RX errors 0 dropped 18926930 overruns 0 frame 0
TX packets 2057 bytes 926192
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

##### 5. Check if ACI is sending the packets out or the destination is receiving the packets

In case none of the above is applicable in this scenario, take a SPAN capture on the egress port of the leaf switch to eliminate ACI fabric from the troubleshooting path.

Rx counters on the destination can also be useful to eliminate the entire network switches from troubleshooting path as shown in the previous steps for buffering.

## Endpoint flapping

This section explains how to check endpoint flapping briefly. See the following two documents for details:

- "ACI Fabric Endpoint Learning Whitepaper" on [www.cisco.com](http://www.cisco.com)
- "Cisco Live BRKACI-2641 ACI Troubleshooting: Endpoints" on [www.ciscolive.com](http://www.ciscolive.com)

When ACI learns the same MAC or IP address in multiple locations, it will look as if the endpoint is moving even though one of them may be caused by a spoofing device or mis-configuration. Such behavior is referred to as endpoint flapping. In such a scenario, traffic towards the moving/flapping endpoint (MAC address for bridged traffic, IP address for routed traffic) will intermittently fail.

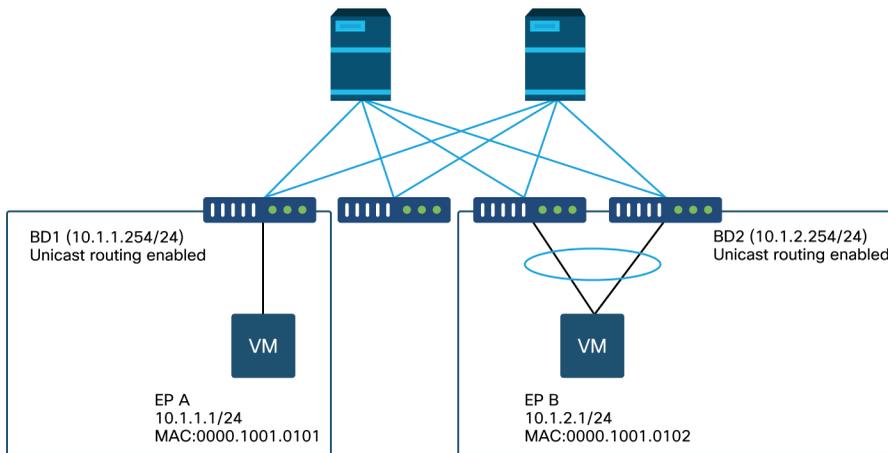
The most effective method to detect endpoint flapping is to use the Enhanced Endpoint Tracker. This app can run as an ACI AppCenter app or as a standalone app on an external server in case it needs to manage a much larger fabric.

### Enhanced Endpoint Tracker

The screenshot displays the ACI AppCenter interface. At the top, the Cisco logo and 'APIC' are visible. Below this is a navigation bar with tabs for 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Apps' tab is currently selected. Below the navigation bar, there are three app cards. The first card is 'ELAM Assistant' by Cisco, with a description: 'Help you perform ELAM(Embedded Logic Analyzer Module) on ACI nodes to capture a single packet at a time and analyze where the packet goes.' It has an 'Open' button and a status indicator of 0. The second card, 'EnhancedEndpointT' by Cisco, is highlighted with a red box. Its description is 'Track endpoint activity within the ACI fabric'. It has an 'Open' button and a status indicator of 0. The third card is 'Network Insights - Resources' by Cisco, with a description: 'Network Insights - Resources is a platform for predictive analytics, correlation and alerting using streaming telemetry data for networking fabrics.' It has an 'Enable' button and a status indicator of 1.

The picture above shows the Enhanced Endpoint Tracker in AppCenter. The following shows an example of how to find flapping endpoints with the Enhanced Endpoint Tracker.

#### Endpoint flapping example



In this example, IP 10.1.2.1 should belong to EP B with MAC 0000.1001.0102. However, an EP X with MAC 0000.1001.9999 is also sourcing traffic with IP 10.1.2.1 due to a misconfig or perhaps IP spoofing.

## Enhanced Endpoint Tracker output – Moves

🔄
10.1.2.1

Actions ▾

Fabric **TK-FAB2** VRF **uni/tn-TK/ctx-VRF1** EPG **uni/tn-TK/ap-APP1/epg-EPG2-3**  
 Local on **pod-1** node **103** interface **eth1/3** encap **vlan-2203** mac **00:00:10:01:99:99**  
 Remotely learned on **3** nodes. ▾

109 Moves
0 Rapid events
0 OffSubnet events
0 Stale events
0 Clear events

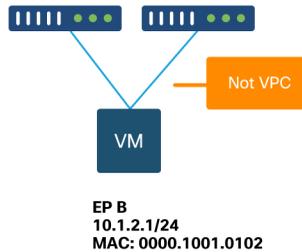
History
🔍 Detailed
➔ Move
⚡ Rapid
📁 OffSubnet
⬆️ Stale
🗑️ Cleared

Time ^	Local Node	Status	Interface	Encap	pcTAG	MAC	EPG
Oct 01 2019 - 15:21:08	103	created	eth1/3	vlan-2203	32773	00:00:10:01:99:99	uni/tn-TK/ap-APP1/epg-EPG2-3
Oct 01 2019 - 15:21:08	(103,104)	created	NSK_VPC_3-4_13	vlan-3134	32774	00:00:10:01:01:02	uni/tn-TK/ap-APP1/epg-EPG2-1
Oct 01 2019 - 15:21:06	103	created	eth1/3	vlan-2203	32773	00:00:10:01:99:99	uni/tn-TK/ap-APP1/epg-EPG2-3
Oct 01 2019 - 15:21:06	(103,104)	created	NSK_VPC_3-4_13	vlan-3134	32774	00:00:10:01:01:02	uni/tn-TK/ap-APP1/epg-EPG2-1
Oct 01 2019 - 15:21:04	103	created	eth1/3	vlan-2203	32773	00:00:10:01:99:99	uni/tn-TK/ap-APP1/epg-EPG2-3
Oct 01 2019 - 15:21:04	(103,104)	created	NSK_VPC_3-4_13	vlan-3134	32774	00:00:10:01:01:02	uni/tn-TK/ap-APP1/epg-EPG2-1
Oct 01 2019 - 15:21:02	103	created	eth1/3	vlan-2203	32773	00:00:10:01:99:99	uni/tn-TK/ap-APP1/epg-EPG2-3
Oct 01 2019 - 15:21:02	(103,104)	created	NSK_VPC_3-4_13	vlan-3134	32774	00:00:10:01:01:02	uni/tn-TK/ap-APP1/epg-EPG2-1
Oct 01 2019 - 15:21:00	103	created	eth1/3	vlan-2203	32773	00:00:10:01:99:99	uni/tn-TK/ap-APP1/epg-EPG2-3

The Enhanced Endpoint Tracker shows when and where IP 10.1.2.1 was learned. As shown in the screenshot above, 10.1.2.1 is flapping between two endpoints with MAC 0000.1001.0102 (expected) and 0000.1001.9999 (not expected). This will cause a reachability issue towards IP 10.1.2.1 because when it's learned on the wrong MAC address, the packet will be sent to a wrong device via the wrong interface. To resolve this, take steps to prevent the unexpected VM from sourcing traffic with an inappropriate IP address.

The following shows a typical example of endpoint flapping due to an inappropriate configuration.

Topology example that could cause endpoint flapping



When a server or VM is connected to ACI leaf nodes via two interfaces without a VPC, the server needs to use Active/Standby NIC teaming. Otherwise, the packets are load balanced to both uplinks and it would look as if the endpoints are flapping between two interfaces from the ACI leaf switch perspective. In this case, Active/Standby or equivalent NIC teaming mode is required or just use a VPC on the ACI side.

# Interface drops

This chapter describes how to check major counters related to ingress interface drop.

## Hardware drop counter types

On Nexus 9000 switches running in ACI mode, there are three major hardware counters on the ACI for ingress interface drops.

### Forward

Major reasons this may happen are following:

- **SECURITY\_GROUP\_DENY**: A drop because of missing contracts to allow the communication.
- **VLAN\_XLATE\_MISS**: A drop because of inappropriate VLAN. For example, a frame enters the fabric with an 802.1Q VLAN 10. If the switch has VLAN 10 on the port, it will inspect the contents and make a forwarding decision based on the destination MAC. However, if VLAN 10 is not allowed on the port, it will drop it and label it as a **VLAN\_XLATE\_MISS**.
- **ACL\_DROP**: A drop because of SUP-TCAM. SUP-TCAM in ACI switches contains special rules to be applied on top of the normal L2/L3 forwarding decision. Rules in SUP-TCAM are built-in and not user configurable. The objective of SUP-TCAM rules is mainly to handle some exceptions or some control plane traffic and not intended to be checked or monitored by users. When a packet is hitting SUP-TCAM rules and the rule is to drop the packet, the dropped packet is counted as **ACL\_DROP** and it will increment the forward drop counter.

Forward drops are essentially packets dropped for a valid known reason. They can generally be ignored and will not cause performance penalties, unlike real data traffic drops.

### Error

When the switch receives an invalid frame, it is dropped as an error. Examples of this include frames with FCS or CRC errors. See the later section "CRC – FCS – cut-through switching" for more details.

### Buffer

When a switch receives a frame, and there are no buffers available for either ingress or egress, the frame will be dropped with 'Buffer'. This typically hints at congestion somewhere in the network. The link that is showing the fault could be full, or the link containing the destination may be congested.

## Viewing drop stats in CLI

If faults are noted, or there is a need to check packet drops on interfaces using the CLI, the best way to do this is by viewing the platform counters in hardware. Not all counters are shown using 'show interface'. The three major drop reasons can only be viewed using the platform counters. In order to view these, perform these steps:

### Leaf

SSH to the leaf and run these commands. This example is for ethernet 1/31.

```

ACI-LEAF# vsh_lc;
module-1# show platform internal counters port 31;
Stats for port 31;
(note: forward drops includes sup redirected packets too);
IF          LPort          Input              Output;
           Pkts      Bytes      Pkts      Bytes;
eth-1/31   31  Total    400719   286628225  2302918   463380330;
           Unicast  306610   269471065  453831    40294786;
           Multicast  0         0         1849091   423087288;
           Flood     56783    8427482    0         0;
Total Drops  37327          0;
Buffer         0          0;
Error         0          0;
Forward       37327;
           LB         0;
           AFD RED          0;
...

```

## Spine

A fixed spine (N9K-C9332C and N9K-C9364C) can be checked using the same method as the leaf switches.

For a modular spine (N9K-C9504 etc.), the linecard must be attached to before the platform counters can be viewed. SSH to the spine and run these commands. This example is for ethernet 2/1.

```

ACI-SPINE# vsh
ACI-SPINE# attach module 2
module-2# show platform internal counters port 1
Stats for port 1
(note: forward drops include sup redirected packets too)
IF          LPort          Input          Output
           Packets    Bytes          Packets    Bytes
eth-2/1    1 Total      85632884 32811563575 126611414 25868913406
           Unicast  81449096 32273734109 104024872 23037696345
           Multicast 3759719 487617769 22586542 2831217061
           Flood      0         0           0         0
           Total Drops 0         0           0         0
           Buffer      0         0           0         0
           Error      0         0           0         0
           Forward    0         0           0         0
           LB          0
           AFD RED          0
...

```

Queuing stats counters are shown using 'show queuing interface'. This example is for ethernet 1/5.

```

ACI-LEAF# show queuing interface ethernet 1/5
=====
Queuing stats for ethernet 1/5
=====
Qos Class level1
=====
Rx Admit Pkts : 0           Tx Admit Pkts : 0
Rx Admit Bytes: 0           Tx Admit Bytes: 0
Rx Drop Pkts  : 0           Tx Drop Pkts  : 0
Rx Drop Bytes : 0           Tx Drop Bytes : 0
=====
Qos Class level2

```

```
=====  
Rx Admit Pkts : 0                Tx Admit Pkts : 0  
Rx Admit Bytes: 0                Tx Admit Bytes: 0  
Rx Drop Pkts  : 0                Tx Drop Pkts  : 0  
Rx Drop Bytes : 0                Tx Drop Bytes : 0  
=====
```

Qos Class level3

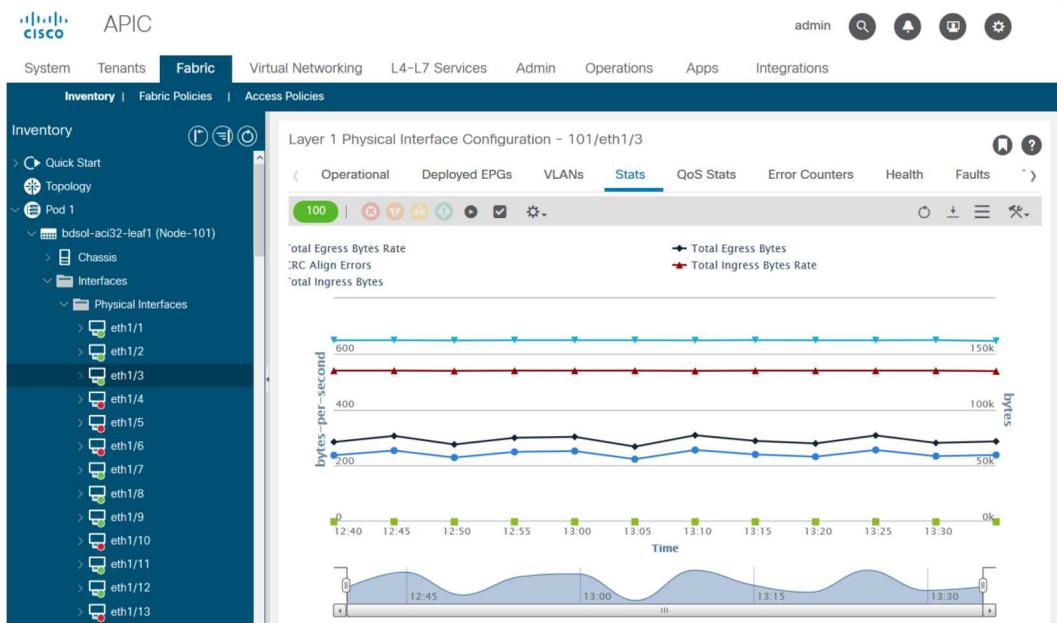
```
=====  
Rx Admit Pkts : 1756121          Tx Admit Pkts : 904909  
Rx Admit Bytes: 186146554        Tx Admit Bytes: 80417455  
Rx Drop Pkts  : 0                Tx Drop Pkts  : 22  
Rx Drop Bytes : 0                Tx Drop Bytes : 3776  
=====
```

...

## Viewing statistics in GUI

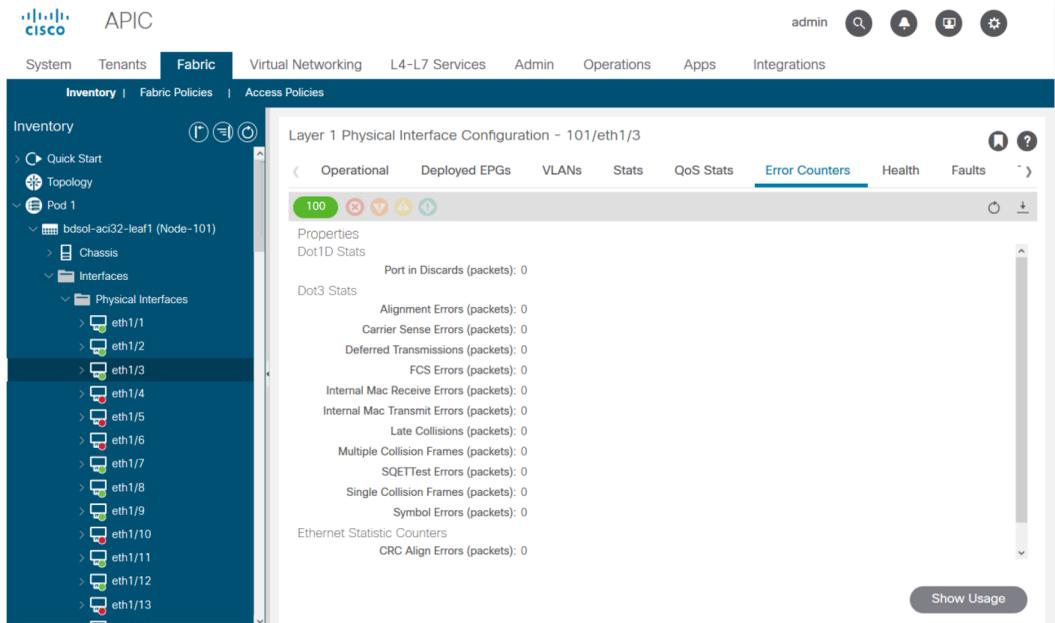
The location is 'Fabric > Inventory > Leaf/Spine > Physical interface > Stats'.

GUI interface statistics



The error statistics can be seen in the same place:

### GUI interface errors



And finally, the GUI can display QoS stats per interface:

## GUI interface QoS counters

The screenshot shows the Cisco APIC GUI interface for QoS counters. The left-hand navigation pane is open, showing the hierarchy: Inventory > Pod 1 > bdsol-aci32-leaf1 (Node-101) > Interfaces > Physical Interfaces > eth1/3. The main panel displays the configuration for Layer 1 Physical Interface Configuration - 101/eth1/3. The 'QoS Stats' tab is selected, showing a table of Rx Counts for various classes.

Class	Rx Counts				#
	Admit Bytes	Admit Packets	Drop Bytes	Drop Packets	
level3	708675836054	10353168921	0	0	66345
level2	0	0	0	0	0
level1	0	0	0	0	0
policy-plane	1713394062	23810156	612868452	8543387	0
control-plane	515330151	5939396	0	0	94521
span	0	0	0	0	0
level6	0	0	0	0	0
level5	0	0	0	0	0
level4	0	0	0	0	0

## CRC – FCS – cut-through switching

### What is cyclic redundancy check (CRC)?

CRC is a polynomial function on the frame which returns a 4B number in Ethernet. It will catch all single bit errors and a good percentage of double bit errors. It is thus meant to ensure that the frame was not corrupted in transit. If the CRC error counter is increasing, it means that when the hardware ran the polynomial function on the frame, the result was a 4B number which differed from the 4B number found on the frame itself. Frames can get corrupted due to several reasons such as duplex mismatch, faulty cabling, and broken hardware. However, some level of CRC errors should be expected and the standard allows up-to  $10^{-12}$  bit-error-rate on Ethernet (1 bit out of  $10^{12}$  can flip).

### Store-and-forward vs cut-through switching

Both store-and-forward and cut-through Layer 2 switches base their forwarding decisions on the destination MAC address of data packets. They also learn MAC addresses as they examine the source MAC (SMAC) fields of packets as stations communicate with other nodes on the network.

A store-and-forward switch makes a forwarding decision on a data packet after it has received the entire frame and checked its integrity. A cut-through switch engages in the forwarding process soon after it has examined the destination MAC (DMAC) address of an incoming frame. However, a cut-through switch must wait until it has viewed the entire packet before performing the CRC check. That means that by the time CRC is validated, the packet has already been forwarded and cannot be dropped if it fails the check.

Traditionally, most network devices used to operate based on store-and-forward. Cut-through switching technologies tend to get used in high speed networks that demand low latency forwarding.

All devices in ACI fabric are doing cut-through switching.

### Stomping

Packets with a CRC error necessitate a drop. If the frame is being switched in a cut-through path, CRC validation happens after the packet is already forwarded. As such, the only option is to stomp the Ethernet Frame Check Sequence (FCS). **Stomping a frame involves setting the FCS to a known value that does not pass a CRC check.** Because of this, one bad frame that fails CRC could show up as a CRC on every interface it traverses, until it reaches a store-and-forward switch which will drop it.

### ACI and CRC: look for faulty interfaces

- If a leaf sees CRC errors on a downlink port, it is mostly a problem on the downlink SFP or with components on the external device/network.

- If a spine sees CRC errors, it is mostly a problem on that local port, SFP, Fiber or Neighbor SFP. CRC failing packets from leaf downlinks are not stomped to the spines. As if its headers are readable, it is VXLAN encapsulated and new CRC will be computed. If the headers were not readable from frame corruption, the packet would be dropped.
- If a leaf sees CRC errors on fabric links, it can either be:
  - An issue on the local fiber/SFP pair, the spine's ingress fiber, or the SFP pair.
  - A stomped frame making its way through the fabric.

### Stomping: troubleshoot stomping

- Look for interfaces with FCS errors on the fabric. Since FCS occurs local to a port, it is most likely the fiber or SFP on either end.
- CRC errors on 'show interface' output reflects the total FCS+Stomp value.

Look at an example:

Check on a port with the command in vsh\_lc: 'show platform internal counter port <X>'.

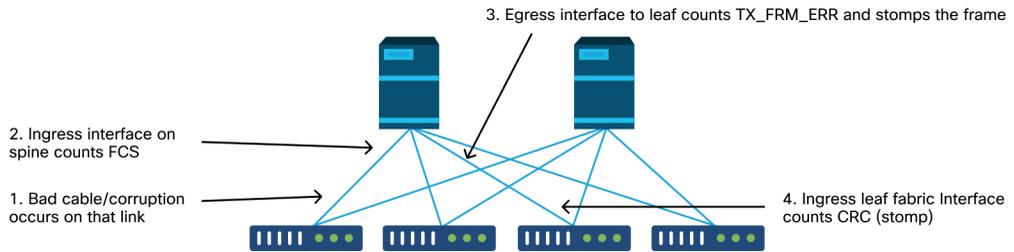
In this command 3 values are important:

- RX\_FCS\_ERR - FCS failure.
- RX\_CRCERR - Received stomped CRC error frame.
- TX\_FRM\_ERROR - Transmitted stomped CRC error frame.

```
module-1# show platform internal counters port 1 | egrep ERR
RX_FCS_ERR          0      ---- Real error local between the devices and its direct neighbor
RX_CRCERR           0      ---- Stomped frame --- so likely stomped by underlying devices and generated further down the network
TX_FRM_ERROR        0      ---- Packet received from another interface that was stomped on Tx direction
```

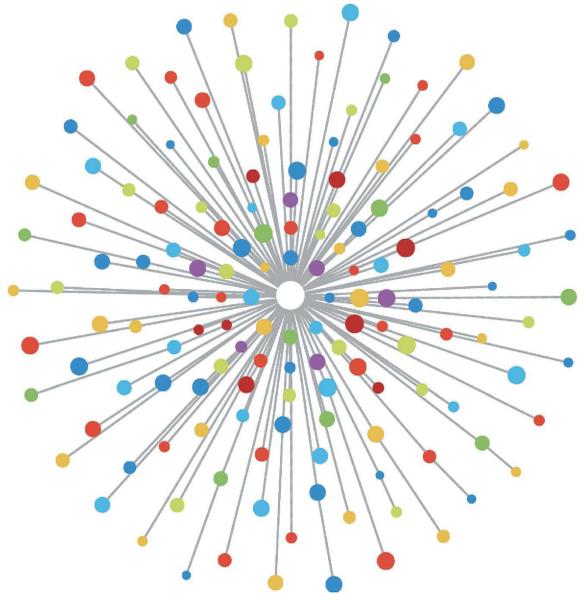
Example topology:

CRC stomp troubleshooting scenario



If a corrupted link generates a large number of corrupted frames, those frames could be flooded to all other leaf nodes and it is very possible to find CRC on the ingress of fabric uplinks of most leaf nodes in the fabric. Those would likely all come from a single corrupted link.

# External forwarding



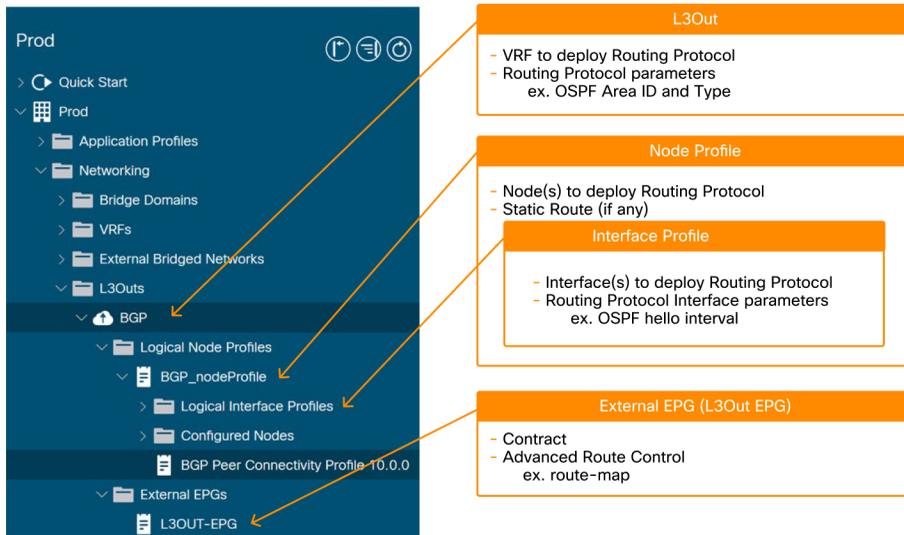
# Overview

In order for traffic to be routed outside of the fabric, Cisco ACI uses the concept of an L3Out which is configured at the tenant level. This section further explores the mechanisms of an L3Out and how to configure and troubleshoot it.

## L3Out components

The following picture illustrates the major building blocks required to configure an L3 Outside (L3Out)

Major components of a L3Out



### 1 Root of L3Out:

- Select a routing protocol to deploy (such as OSPF, BGP).
- Select a VRF to deploy the routing protocol.
- Select an L3Out Domain to define available leaf interfaces and VLAN for the L3Out.

### 2 Node Profile:

- Select leaf switches to deploy the routing protocol. These are typically known as 'Border Leaf Switches' (BL).
- Configure Router-ID (RID) for the routing protocol on each border leaf. Unlike a normal router, ACI does not automatically assign Router-ID based on an IP address on the switch.
- Configure a static route.

### 3 Interface profile:

- Configure leaf interfaces to run the routing protocol.  
i.e. Interface type (SVI, routed-port, sub-interface), interface ID and IP addresses etc.
- Select a policy for interface level routing protocol parameters (such as hello interval).

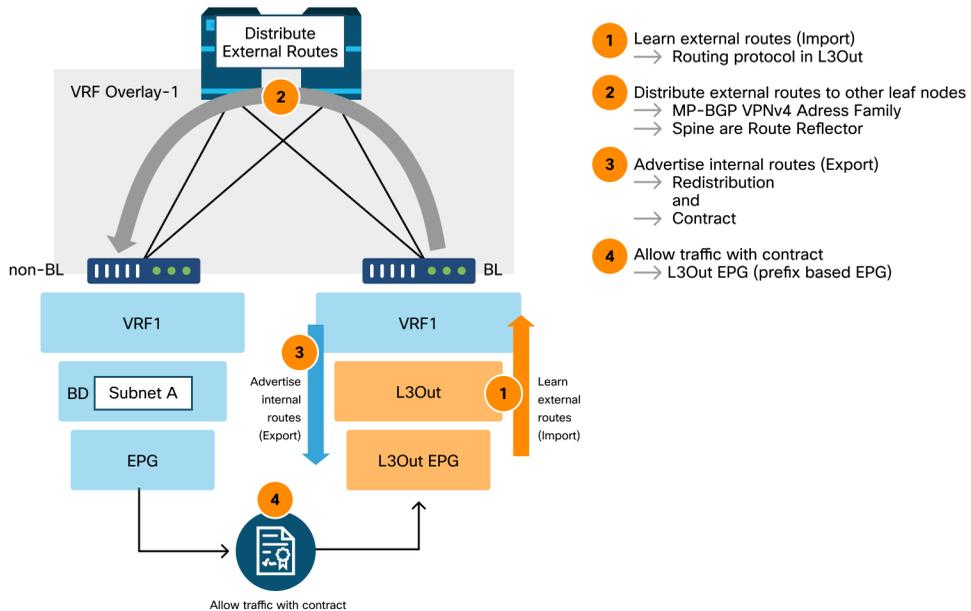
### 4 External EPG (L3Out EPG):

- An 'External EPG' is a hard requirement to deploy all policy tied to the L3Out, such as IP addresses or SVIs to establish neighbors. Details on how to use external EPGs will be covered later.

## External routing

The following diagram shows the high-level operation involved for external routing.

High-level external routing flow



- 1 The BL(s) will establish routing protocol adjacencies with external routers.
- 2 Route prefixes are received from external routers and are injected in MP-BGP as the VPNv4 address-family path.
  - At a minimum, two spine nodes must be configured as BGP route reflectors to reflect external routes to all leaf nodes.

- 3 Internal prefixes (BD subnets) and/or prefixes received from other L3Out must be explicitly redistributed into the routing protocol to be advertised to the external router.
- 4 Security enforcement: an L3Out contains at least one L3Out EPG. A contract must be consumed or provided on the L3Out EPG (also called l3extInstP from the class name) to allow traffic in/out of the L3Out.

## L3Out EPG configuration options

In the L3Out EPG section, subnets can be defined with a series of 'Scope' and 'Aggregate' options as illustrated below:

An L3Out subnet being defined including 'scope' definition

### Create Subnet



IP Address:   
address/mask

Name:

- scope:
- Export Route Control Subnet
  - Import Route Control Subnet
  - External Subnets for the External EPG
  - Shared Route Control Subnet
  - Shared Security Import Subnet

BGP Route Summarization Policy:

- aggregate:
- Aggregate Export
  - Aggregate Import
  - Aggregate Shared Routes

Route Control Profile:

Name	Direction

Cancel

Submit

'Scope' options:

- **Export Route Control Subnet:** This scope is to advertise (export) a subnet from ACI to outside via the L3Out. Although this is mainly for Transit Routing, this could also be used to advertise a BD subnet as described in the "ACI BD subnet advertisement" section.
- **Import Route Control Subnet:** This scope is about learning (importing) an external subnet from the L3Out. By default, this scope is disabled, hence it's greyed out, and a border leaf (BL) learns any routes from a routing protocol. This scope can be enabled when it needs to limit external routes learned via OSPF and BGP. This is not supported for EIGRP. To use this scope, 'Import Route Control Enforcement' needs to be enabled first on a given L3Out.
- **External Subnets for the External EPG (import-security):** This scope is used to allow packets with the configured subnet from or to the L3Out with a contract. It classifies a packet into the configured L3Out EPG based on the subnet so that a contract on the L3Out EPG can be applied to the packet. This scope is a Longest Prefix Match instead of an exact match like other scopes for routing table. If 10.0.0.0/16 is configured with 'External Subnets for the External EPG' in L3Out EPG A, any packets with IP in that subnet, such as 10.0.1.1, will be classified into the L3Out EPG A to use a contract on it. This does not mean 'External Subnets for the External EPG' scope installs a route 10.0.0.0/16 in a routing table. It will create a different internal table to map a subnet to an EPG (pcTag) purely for a contract. It does not have any effects on routing protocol behaviors. This scope is to be configured on a L3Out that is learning the subnet.
- **Shared Route Control Subnet:** This scope is to leak an external subnet to another VRF. ACI uses MP-BGP and Route Target to leak an external route from one VRF to another. This scope creates a prefix-list with the subnet, which is used as a filter to export/import routes with route target in MP-BGP. This scope is to be configured on a L3Out that is learning the subnet in the original VRF.
- **Shared Security Import Subnet:** This scope is used to allow packets with the configured subnet when the packets are moving across VRFs with a L3Out. A route in a routing table is leaked to another VRF with 'Shared Route Control

Subnet' as mentioned above. However, another VRF has yet to know which EPG the leaked route should belong to. The 'Shared Security Import Subnet' informs another VRF of the L3Out EPG which the leaked route belongs to. Hence, this scope can be used only when 'External Subnets for the External EPG' is also used, otherwise the original VRF doesn't know which L3Out EPG the subnet belongs to either. This scope is also the Longest Prefix Match.

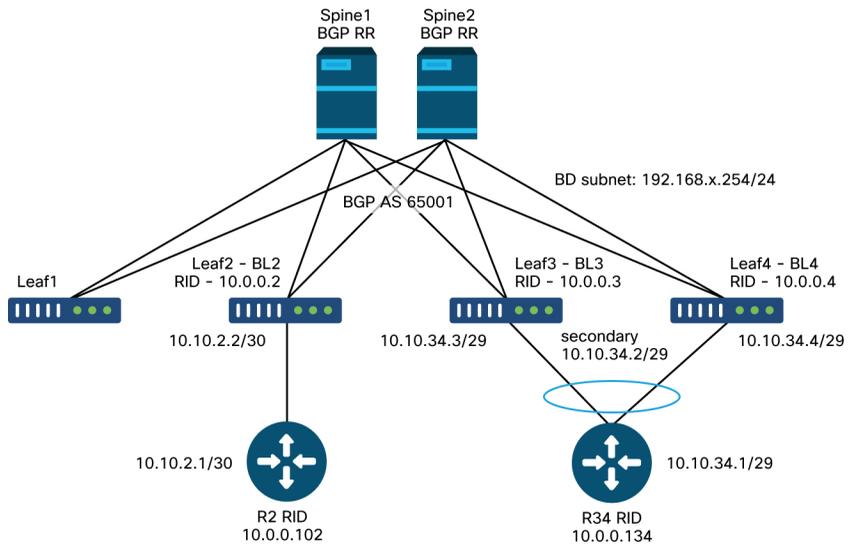
'Aggregate' options:

- **Aggregate Export:** This option can be used only for 0.0.0.0/0 with 'Export Route Control Subnet'. When both 'Export Route Control Subnet' and 'Aggregate Export' are enabled for 0.0.0.0/0; it creates a prefix-list with '0.0.0.0/0 le 32' which matches any subnets. Hence, this option can be used when a L3Out needs to advertise (export) any routes to the outside. When more granular aggregation is required, Route Map/Profile with an explicit prefix-list can be used.
- **Aggregate Import:** This option can be used only for 0.0.0.0/0 with 'Import Route Control Subnet'. When both 'Import Route Control Subnet' and 'Aggregate Import' are enabled for 0.0.0.0/0, it creates a prefix-list with '0.0.0.0/0 le 32' which matches any subnets. Hence, this option can be used when a L3Out needs to learn (import) any routes from outside. However, the same thing can be accomplished by disabling 'Import Route Control Enforcement' which is the default. When more granular aggregation is required, Route Map/Profile with an explicit prefix-list can be used.
- **Aggregate Shared Routes:** This option can be used for any subnets with 'Shared Route Control Subnet'. When both 'Shared Route Control Subnet' and 'Aggregate Shared Routes' are enabled for 10.0.0.0/8 as an example, it creates a prefix-list with '10.0.0.0/8 le 32' which matches 10.0.0.0/8, 10.1.0.0/16 and so on.

## L3Out topology used in this section

The following topology will be used in this section:

L3Out topology



External subnet range: 172.16.x.0/24

# Adjacencies

This section explains how to troubleshoot and verify routing protocol adjacencies on L3Out interfaces.

Below are a few parameters to check that will be applicable for all ACI external routing protocols:

- **Router ID:** In ACI, each L3Out needs to use the same Router ID in the same VRF on the same leaf even if routing protocols are different. Also, only one of those L3Outs on the same leaf can create a loopback with the Router ID, which is typically BGP.
- **MTU:** Although the hard requirement of MTU is only for OSPF adjacency, it is recommended to match MTU for any routing protocols to ensure any jumbo packets used for route exchange/updates can be transmitted without fragmentation, as most of control plane frames will be sent with the DF (don't fragment) bit set, which will drop the frame if its size exceeds the maximum MTU of the interface.
- **MP-BGP Router Reflector:** Without this, the BGP process will not start on leaf nodes. Although, this is not required for OSPF or EIGRP just to establish a neighbor, it is still required for BLs to distribute external routes to other leaf nodes.
- **Faults:** Always be sure to check faults during and after configuration is complete.

## BGP

This section uses an example of an eBGP peering between the loopback on BL3, BL4, and R34 from the topology in the Overview section. The BGP AS on R34 is 65002.

Verify the following criteria when establishing a BGP adjacency.

- Local BGP AS number (ACI BL side).

### Peer Connectivity Profile – Local-AS

Peer Connectivity Profile - BGP Peer Connectivity Profile 10.10.34.1

Policy    Faults    History

Properties

Remote Autonomous System Number: 65002

Local-AS Number Config:

Local-AS Number:

This value must not match the MP-BGP RR policy

Admin State: Disabled Enabled

The BGP AS number of a user L3Out will automatically be the same as the BGP AS for the infra-MP-BGP that is configured in the BGP Route Reflector policy. The 'Local AS' configuration in the BGP Peer Connectivity Profile is not required unless one needs to disguise the ACI BGP AS to the outside world. This means external routers should point to the BGP AS configured in the BGP Route Reflector.

NOTE – The scenario where Local AS configuration is required is the same as the standalone NX-OS 'local-as' command.

- Remote BGP AS number (external side)

## Peer Connectivity Profile – Remote AS

Peer Connectivity Profile - BGP Peer Connectivity Profile 10.10.34.1

Policy Faults History

Properties

Remote Autonomous System Number: 65002

Local-AS Number Config:

Local-AS Number:

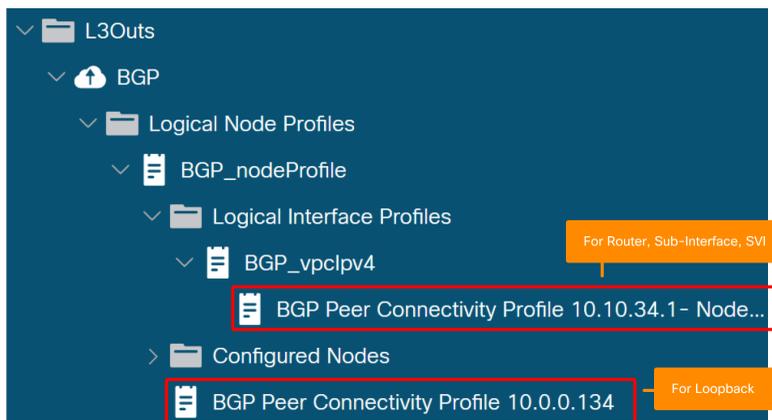
This value must not match the MP-BGP RR policy

Admin State: Disabled Enabled

The Remote BGP AS number is required only for eBGP where the neighbor's BGP AS is different from ACI BGP AS.

- Source IP for BGP peer session.

## L3Out – BGP Peer Connectivity Profile



ACI supports sourcing a BGP session from the loopback interface on top of a typical ACI L3Out interface type (routed, sub-interface, SVI).

When a BGP session needs to be sourced from a loopback, configure the BGP Peer Connectivity Profile under the Logical **Node** Profile.

When the BGP session needs to be sourced from a routed/sub-interface/SVI, configure the BGP Peer Connectivity Profile under the Logical **Interface** Profile.

- BGP peer IP reachability.

### Logical Node Profile – Node Association

The screenshot displays the Cisco APIC interface for configuring a Logical Node Profile. The left-hand navigation pane shows the hierarchy: Prod > L3Outs > BGP > Logical Node Profiles > BGP\_nodeProfile. The main configuration area is titled 'Node Association' and shows the following details:

- Node ID:** topology/pod-1/node-103
- Router ID:** 10.0.0.3
- Use Router ID as Loopback Address:**  (Note: This setting will be ignored if loopback addresses are defined in the table below.)
- Loopback Addresses:** A table with one header row 'IP' and no data rows. A message below states: 'No items have been found. Select Actions to create a new item.'
- Static Routes:** A table with three columns: IP Address, Track Policy, and Next Hop IP. It contains one entry:
 

IP Address	Track Policy	Next Hop IP
10.0.0.134/32		10.10.34.1

When the BGP peer IPs are loopbacks, make sure the BL and the external router have reachability to the peer's IP address. Static routes or OSPF can be used to gain reachability to the peer IPs.

### BGP CLI Verification (eBGP with loopback example)

The CLI outputs for the following steps are collected from BL3 in the topology from the Overview section.

## 1. Check if the BGP session is established

'State/PfxRcd' in the following CLI output means the BGP session is established.

```
f2-leaf3# show bgp ipv4 unicast summary vrf Prod:VRF1
BGP summary information for VRF Prod:VRF1, address family IPv4 Unicast
BGP router identifier 10.0.0.3, local AS number 65001

Neighbor      V   AS  MsgRcvd  MsgSent   TblVer  InQ  OutQ  Up/Down  State/PfxRcd
10.0.0.134    4 65002     10       10        10    0    0 00:06:39 0
```

If the 'State/PfxRcd' shows Idle or Active, BGP packets are not being exchanged with the peer yet. In such a scenario, check the following and move on to the next step.

- Ensure the external router is pointing to the ACI BGP AS correctly (local AS number 65001).
- Ensure the ACI BGP Peer Connectivity Profile is specifying the correct neighbor IP from which the external router is sourcing the BGP session (10.0.0.134).
- Ensure the ACI BGP Peer Connectivity Profile is specifying the correct neighbor AS of the external router (Remote Autonomous System Number in GUI which shows up in CLI as AS 65002).

## 2. Check BGP Neighbor details (BGP Peer Connectivity Profile)

The following command shows the parameters that are key for BGP neighbor establishment.

- Neighbor IP: 10.0.0.134.
- Neighbor BGP AS: remote AS 65002.
- Source IP: Using loopback3 as update source.
- eBGP multi-hop: External BGP peer might be upto 2 hops away.

```
f2-leaf3# show bgp ipv4 unicast neighbors vrf Prod:VRF1
BGP neighbor is 10.0.0.134, remote AS 65002, ebgp link, Peer index 1
BGP version 4, remote router ID 10.0.0.134
BGP state = Established, up for 00:11:18
Using loopback3 as update source for this peer
External BGP peer might be upto 2 hops away
```

```

...

For address family: IPv4 Unicast
...
Inbound route-map configured is permit-all, handle obtained
Outbound route-map configured is exp-l3out-BGP-peer-3047424, handle obtained
Last End-of-RIB received 00:00:01 after session start
Local host: 10.0.0.3, Local port: 34873
Foreign host: 10.0.0.134, Foreign port: 179
fd = 64

```

Once the BGP peer is established correctly, the 'Local host' and 'Foreign host' appear at the bottom of the output.

### 3. Check IP reachability for the BGP peer

```

f2-leaf3# show ip route vrf Prod:VRF1
10.0.0.3/32, ubest/mbest: 2/0, attached, direct
  *via 10.0.0.3, lo3, [0/0], 02:41:46, local, local
  *via 10.0.0.3, lo3, [0/0], 02:41:46, direct
10.0.0.4/32, ubest/mbest: 1/0
  *via 20.0.32.68%overlay-1, [1/0], 02:38:04, bgp-65001, internal, tag 65001
10.0.0.134/32, ubest/mbest: 1/0
  *via 10.10.34.1, vln27, [1/0], 02:41:46, static <--- neighbor IP reachability via static route
10.10.34.0/29, ubest/mbest: 2/0, attached, direct
  *via 10.10.34.3, vln27, [0/0], 02:41:46, direct
  *via 10.10.34.2, vln27, [0/0], 02:41:46, direct
10.10.34.2/32, ubest/mbest: 1/0, attached
  *via 10.10.34.2, vln27, [0/0], 02:41:46, local, local
10.10.34.3/32, ubest/mbest: 1/0, attached
  *via 10.10.34.3, vln27, [0/0], 02:41:46, local, local

```

Ensure ping to the neighbor IP works from ACI BGP's source IP.

```

f2-leaf3# iping 10.0.0.134 -V Prod:VRF1 -S 10.0.0.3
PING 10.0.0.134 (10.0.0.134) from 10.0.0.3: 56 data bytes
64 bytes from 10.0.0.134: icmp_seq=0 ttl=255 time=0.571 ms
64 bytes from 10.0.0.134: icmp_seq=1 ttl=255 time=0.662 ms

```

#### 4. Check the same thing on the external router

The following is an example of configuration on the external router (standalone NX-OS).

```

router bgp 65002
vrf f2-bgp
  router-id 10.0.0.134
  neighbor 10.0.0.3
    remote-as 65001
    update-source loopback134
  ebgp-multihop 2
  address-family ipv4 unicast
  neighbor 10.0.0.4
    remote-as 65001
    update-source loopback134
  ebgp-multihop 2
  address-family ipv4 unicast

interface loopback134
vrf member f2-bgp
ip address 10.0.0.134/32

interface Vlan2501
no shutdown
vrf member f2-bgp
ip address 10.10.34.1/29

vrf context f2-bgp
ip route 10.0.0.0/29 10.10.34.2

```

#### 5. Additional Step – tcpdump

On ACI leaf nodes, the tcpdump tool can sniff the 'kpm\_inb' CPU interface to confirm if the protocol packets reached the leaf CPU. Use L4 port 179 (BGP) as a filter.

```

f2-leaf3# tcpdump -ni kpm_inb port 179
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes
20:36:58.292903 IP 10.0.0.134.179 > 10.0.0.3.34873: Flags [P.], seq 3775831990:3775832009, ack 807595300, win
3650, length 19: BGP, length: 19
20:36:58.292962 IP 10.0.0.3.34873 > 10.0.0.134.179: Flags [.], ack 19, win 6945, length 0
20:36:58.430418 IP 10.0.0.3.34873 > 10.0.0.134.179: Flags [P.], seq 1:20, ack 19, win 6945, length 19: BGP,
length: 19
20:36:58.430534 IP 10.0.0.134.179 > 10.0.0.3.34873: Flags [.], ack 20, win 3650, length 0

```

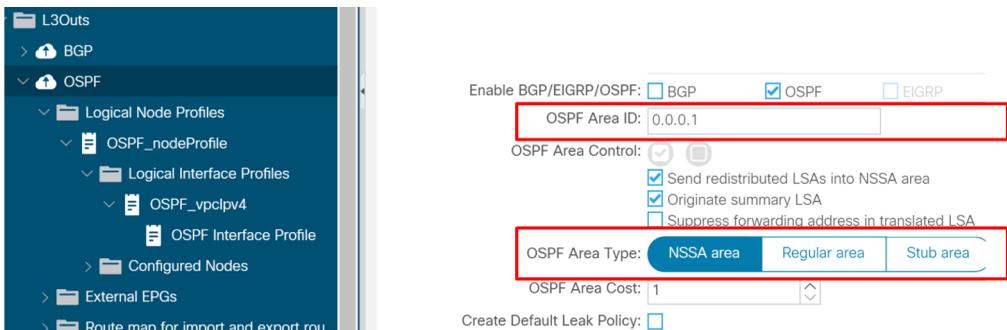
## OSPF

This section uses an example of OSPF neighborships between BL3, BL4, and R34 from the topology in Overview section with OSPF AreaID 1 (NSSA).

The following are the common criteria to check for OSPF adjacency establishment.

- OSPF Area ID and Type

### L3Out – OSPF Interface Profile – Area ID and Type



Just like any routing device, OSPF Area ID and Type need to match on both neighbors. Some ACI specific limitations on OSPF Area ID configurations include:

- One L3Out can have only one OSPF Area ID.
- Two L3Outs can use the same OSPF Area ID in the same VRF only when they are on two different leaf nodes.

Although the OSPF ID does not need to be backbone 0, in the case of Transit Routing it is required between two OSPF L3Outs on the same leaf; one of them must use OSPF Area 0 because any route exchange between OSPF areas must be through OSPF Area 0.

- MTU

## Logical Interface Profile – SVI

Logical Interface Profile - OSPF\_vpclpv4

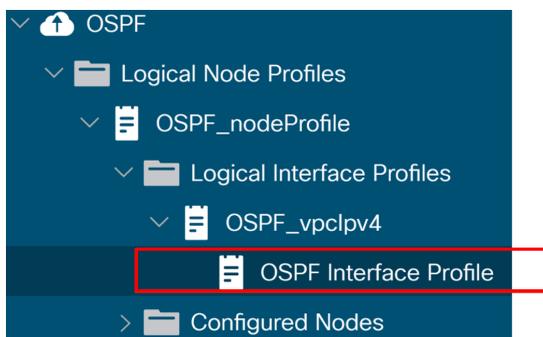


Path	Side A IP	Side B IP	Secondary IP Address	IP Address	MAC Address	MTU (bytes)	Encap	Encap Scope
Pod-1/Node-103-104/N9K_VPC_3-4_13	10.10.34.3/29	10.10.34.4/29	10.10.34.2/29	0.0.0.0	00:22:BD:F8:19:FF	9000	vlan-2502	Local

The default MTU on ACI is 9000 bytes, instead of 1500 bytes, which is typically the default used on traditional routing devices. Ensure the MTU matches with the external device. When OSPF neighbor establishment fails due to MTU, it gets stuck at EXCHANGE/DROTHER.

- IP Subnet mask. OSPF requires the neighbor IP to use the same subnet mask.
- OSPF Interface Profile.

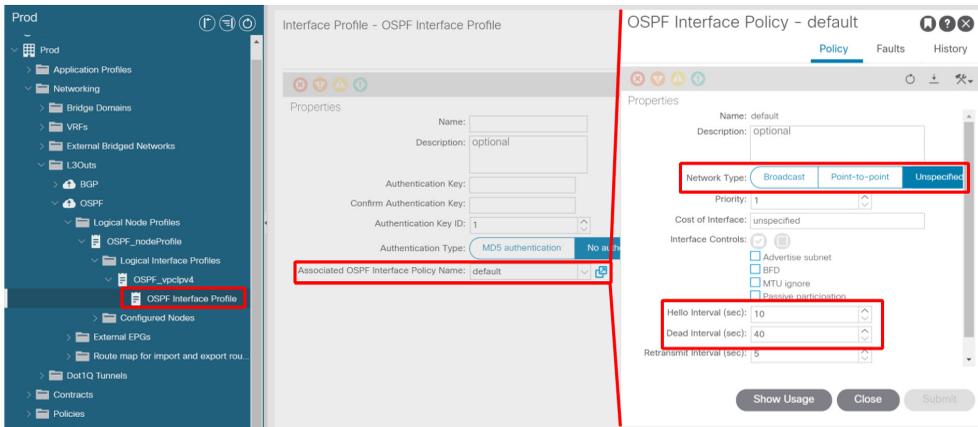
## OSPF Interface Profile



This is equivalent to 'ip router ospf <tag> area <area id>' on a standalone NX-OS config to enable OSPF on the interface. Without this, the leaf interfaces will not join OSPF.

- OSPF Hello / Dead Timer, Network Type

### OSPF Interface Profile – Hello / Dead timer and Network Type



### OSPF Interface Policy details

#### Create OSPF Interface Policy

Name: OSPFIntPolicy

Description: optional

Network Type: Broadcast Point-to-point Unspecified

Priority: 1

Cost of Interface: unspecified

Interface Controls:  Advertise subnet  BFD  MTU ignore  Passive participation

Hello Interval (sec): 10

Dead Interval (sec): 40

Retransmit Interval (sec): 5

Transmit Delay (sec): 1

OSPF requires the Hello and Dead Timers to match on each neighbor device. These are configured in the OSPF Interface Profile.

Ensure the OSPF Interface Network Type matches with the external device. When the external device is using type Point-to-Point, the ACI side needs to explicitly configure Point-to-Point as well. These are also configured in the OSPF Interface Profile.

### OSPF CLI verification

The CLI outputs in the following steps are collected from BL3 in the topology from "Overview" section.

#### 1. Check OSPF neighbor status

If the 'State' is 'FULL' in the following CLI, the OSPF neighbor is established correctly. Otherwise, move on to the next step to check parameters.

```
f2-leaf3# show ip ospf neighbors vrf Prod:VRF2
OSPF Process ID default VRF Prod:VRF2
Total number of neighbors: 2
Neighbor ID      Pri State           Up Time  Address      Interface
10.0.0.4         1 FULL/DR         00:47:30 10.10.34.4   Vlan28      <--- neighbor with BL4
10.0.0.134      1 FULL/DROTHER   00:00:21 10.10.34.1   Vlan28      <--- neighbor with R34
```

In ACI, BLs will form OSPF neighborships with each other on top of external routers when using the same VLAN ID with an SVI. This is because ACI has an internal flooding domain called L3Out BD (or External BD) for each VLAN ID in the L3Out SVIs.

Note that the VLAN ID 28 is an internal VLAN called PI-VLAN (Platform-Independent VLAN) instead of the actual VLAN (Access Encap VLAN) used on wire. Use the following command to verify the access encap VLAN ('vlan-2502').

```
f2-leaf3# show vlan id 28 extended
VLAN Name                               Encap                               Ports
-----
28 Prod:VRF2:l3out-OSPF:vlan-2502 vxlan-14942176, Eth1/13, Po1
                               vlan-2502
```

One could get the same output via access encap VLAN ID as well.

```
f2-leaf3# show vlan encap-id 2502 extended
```

VLAN Name	Encap	Ports
28 Prod:VRF2:13out-OSPF:vlan-2502	vxlan-14942176, vlan-2502	Eth1/13, Po1

## 2. Check OSPF area

Ensure the OSPF area ID and Type is identical to the neighbors. If the OSPF interface profile is missing, the interface will not join OSPF and it will not show up in the OSPF CLI output.

```
f2-leaf3# show ip ospf interface brief vrf Prod:VRF2
```

```
OSPF Process ID default VRF Prod:VRF2
```

```
Total number of interface: 1
```

Interface	ID	Area	Cost	State	Neighbors	Status
Vlan28	94	0.0.0.1	4	BDR	2	up

```
f2-leaf3# show ip ospf vrf Prod:VRF2
```

```
Routing Process default with ID 10.0.0.3 VRF Prod:VRF2
```

```
...
```

### Area (0.0.0.1)

```
Area has existed for 00:59:14
```

```
Interfaces in this area: 1 Active interfaces: 1
```

```
Passive interfaces: 0 Loopback interfaces: 0
```

#### This area is a NSSA area

```
Perform type-7/type-5 LSA translation
```

```
SPF calculation has run 10 times
```

```
Last SPF ran for 0.001175s
```

```
Area ranges are
```

```
Area-filter in 'exp-ctx-proto-3112960'
```

```
Area-filter out 'permit-all'
```

```
Number of LSAs: 4, checksum sum 0x0
```

### 3. Check OSPF interface details

Ensure interface level parameters meet the requirements for OSPF neighbor establishment such as IP subnet, Network Type, Hello/Dead Timer. Please note the VLAN ID to specify the SVI is PI-VLAN (vlan28)

```
f2-leaf3# show ip ospf interface vrf Prod:VRF2
Vlan28 is up, line protocol is up
  IP address 10.10.34.3/29, Process ID default VRF Prod:VRF2, area 0.0.0.1
  Enabled by interface configuration
  State BDR, Network type BROADCAST, cost 4
  Index 94, Transmit delay 1 sec, Router Priority 1
  Designated Router ID: 10.0.0.4, address: 10.10.34.4
  Backup Designated Router ID: 10.0.0.3, address: 10.10.34.3
  2 Neighbors, flooding to 2, adjacent with 2
  Timer intervals: Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello timer due in 0.000000
  No authentication
  Number of opaque link LSAs: 0, checksum sum 0
```

```
f2-leaf3# show interface vlan28
Vlan28 is up, line protocol is up, autostate disabled
Hardware EtherSVI, address is 0022.bdf8.19ff
Internet Address is 10.10.34.3/29
MTU 9000 bytes, BW 10000000 Kbit, DLY 1 usec
```

### 4. Check IP reachability to the neighbor

Although OSPF Hello packets are Link Local Multicast packets, OSPF DBD packets required for the first OSPF LSDB exchange are unicast. Therefore, unicast reachability also needs to be verified for the OSPF neighborship establishment.

```
f2-leaf3# iping 10.10.34.1 -V Prod:VRF2
PING 10.10.34.1 (10.10.34.1) from 10.10.34.3: 56 data bytes
64 bytes from 10.10.34.1: icmp_seq=0 ttl=255 time=0.66 ms
64 bytes from 10.10.34.1: icmp_seq=1 ttl=255 time=0.653 ms
```

## 5. Check the same on the external router

The following are examples of configurations on the external router (standalone NX-OS)

```
router ospf 1
  vrf f2-ospf
  router-id 10.0.0.134
  area 0.0.0.1 nssa

interface Vlan2502
  no shutdown
  mtu 9000
  vrf member f2-ospf
  ip address 10.10.34.1/29
  ip router ospf 1 area 0.0.0.1
```

Make sure to verify the MTU as well on the physical interface.

## 6. Additional step – tcpdump

On ACI leaf nodes, the user can perform tcpdump on the 'kpm\_inb' CPU interface to verify if the protocol packets have reached the leaf CPU. Although there are multiple filters for OSPF, the IP Protocol Number is the most comprehensive filter.

- IP Protocol Number: proto 89 (IPv4) or ip6 proto 0x59 (IPv6)
- IP address of the neighbor: host <ip>
- OSPF Link Local Mcast IP: host 224.0.0.5 or host 224.0.0.6

```
f2-leaf3# tcpdump -ni kpm_inb proto 89
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes
22:28:38.231356 IP 10.10.34.4 > 224.0.0.5: OSPFv2, Hello, length 52
22:28:42.673810 IP 10.10.34.3 > 224.0.0.5: OSPFv2, Hello, length 52
22:28:44.767616 IP 10.10.34.1 > 224.0.0.5: OSPFv2, Hello, length 52
22:28:44.769092 IP 10.10.34.3 > 10.10.34.1: OSPFv2, Database Description, length 32
22:28:44.769803 IP 10.10.34.1 > 10.10.34.3: OSPFv2, Database Description, length 32
22:28:44.775376 IP 10.10.34.3 > 10.10.34.1: OSPFv2, Database Description, length 112
22:28:44.780959 IP 10.10.34.1 > 10.10.34.3: OSPFv2, LS-Request, length 36
22:28:44.781376 IP 10.10.34.3 > 10.10.34.1: OSPFv2, LS-Update, length 64
22:28:44.790931 IP 10.10.34.1 > 224.0.0.6: OSPFv2, LS-Update, length 64
```

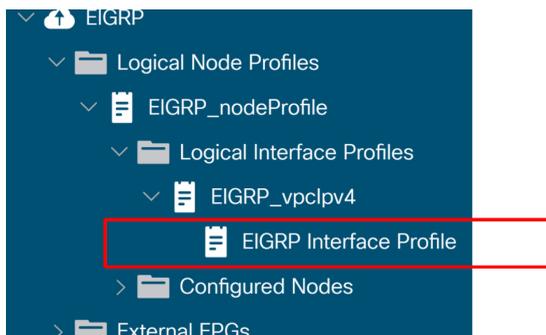
## EIGRP

This section uses an example of EIGRP neighborhood between BL3, BL4 and R34 from the topology in "Overview" section with EIGRP AS 10.

The following are the common criteria for EIGRP adjacency establishment.

- EIGRP AS: a L3Out is assigned one EIGRP AS. This needs to match with the external device.
- EIGRP Interface Profile.

### EIGRP Interface Profile



This is equivalent to the 'ip router eigrp <as>' configuration on a standalone NX-OS device. Without this, the leaf interfaces will not join EIGRP.

- MTU

Although this does not have to match to simply establish the EIGRP neighborhood, the EIGRP topology exchange packets may become larger than the maximum MTU allowed on the interfaces between the peers, and since these packets are not allowed to be fragmented, they are dropped and as a result the EIGRP neighborhood will flap.

## EIGRP CLI Verification

The CLI outputs in the following steps are collected from BL3 in the topology from the "Overview" section.

### 1. Check EIGRP neighbor status

```
f2-leaf3# show ip eigrp neighbors vrf Prod:VRF3
EIGRP neighbors for process 10 VRF Prod:VRF3
H   Address                Interface      Hold  Uptime  SRTT    RTO  Q  Seq
   (sec)                    (ms)          Cnt  Num
0   10.10.34.4              vlan29        14   00:12:58  1    50   0   6   <--- neighbor with BL4
1   10.10.34.1              vlan29        13   00:08:44  2    50   0   4   <--- neighbor with R34
```

In ACI, BLs will form an EIGRP neighborhood with each other on top of external routers when they use the same VLAN ID with SVI. This is because an ACI has an internal flooding domain called L3Out BD (or External BD) for each VLAN ID in L3Out SVIs.

Please note that the VLAN ID 29 is an internal VLAN called PI-VLAN (Platform-Independent VLAN) instead of the actual VLAN (Access Encap VLAN) used on wire. Use the following command to verify the access encap VLAN (vlan-2503).

```
f2-leaf3# show vlan id 29 extended
VLAN Name                Encap                Ports
-----
29   Prod:VRF3:13out-EIGRP:vlan-2503  vxlan-15237052, Eth1/13, Po1
      vlan-2503
```

One could get the same output via access encap VLAN ID as well.

```
f2-leaf3# show vlan encap-id 2503 extended
VLAN Name                Encap                Ports
-----
29   Prod:VRF3:13out-EIGRP:vlan-2503  vxlan-15237052, Eth1/13, Po1
      vxlan-2503
```

## 2. Check EIGRP interface details

Ensure EIGRP is running on the expected interface. If not, check Logical Interface Profile and EIGRP Interface Profile.

```
f2-leaf3# show ip eigrp interfaces vrf Prod:VRF3
EIGRP interfaces for process 10 VRF Prod:VRF3

```

Interface	Peers	Xmit Queue Un/Reliable	Mean SRTT	Pacing Time Un/Reliable	Multicast Flow Timer	Pending Routes
vlan29	2	0/0	1	0/0	50	0

```

Hello interval is 5 sec
Holdtime interval is 15 sec
Next xmit serial: 0
Un/reliable mcasts: 0/2    Un/reliable ucasts: 5/10
Mcast exceptions: 0    CR packets: 0    ACKs suppressed: 2
Retransmissions sent: 2    Out-of-sequence rcvd: 0
Classic/wide metric peers: 2/0

f2-leaf3# show int vlan 29
Vlan29 is up, line protocol is up, autostate disabled
Hardware EtherSVI, address is 0022.bdf8.19ff
Internet Address is 10.10.34.3/29
MTU 9000 bytes, BW 10000000 Kbit, DLY 1 usec

```

## 3. Check the same on the external router

The following the example config on the external router (standalone NX-OS).

```

router eigrp 10
vrf f2-eigrp

interface Vlan2503
no shutdown
vrf member f2-eigrp
ip address 10.10.34.1/29
ip router eigrp 10

```

#### 4. Additional step — tcpdump

On ACI leaf nodes, the user can perform tcpdump on the 'kpm\_inb' CPU interface to confirm if the protocol packets reached the leaf's CPU. Use IP protocol number 88 (EIGRP) as a filter.

```
f2-leaf3# tcpdump -ni kpm_inb proto 88
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes
23:29:43.725676 IP 10.10.34.3 > 224.0.0.10: EIGRP Hello, length: 40
23:29:43.726271 IP 10.10.34.4 > 224.0.0.10: EIGRP Hello, length: 40
23:29:43.728178 IP 10.10.34.1 > 224.0.0.10: EIGRP Hello, length: 40
23:29:45.729114 IP 10.10.34.1 > 10.10.34.3: EIGRP Update, length: 20
23:29:48.316895 IP 10.10.34.3 > 224.0.0.10: EIGRP Hello, length: 40
```

# Route advertisement

This section focusses on the verification and troubleshooting of route advertisement in ACI. Specifically, it looks at examples involving:

- Bridge Domains Subnet Advertisement.
- Transit Route Advertisement.
- Import and Export Route Control.

This section discusses route-leaking as it pertains to shared L3Outs in later sections.

## Bridge domain route advertisement workflow

Before looking at common troubleshooting the user should familiarize themselves with how Bridge Domain advertisement is supposed to work.

BD advertisement, when the BD and L3Out are in the same VRF, involves:

- Having a contract relationship between the L3Out and the internal EPG.
- Associating the L3Out to the Bridge Domain.
- Selecting 'Advertise Externally' on the BD subnet.

In addition, it is also possible to control Bridge Domain advertisement using export route-profiles which prevent the need to associate the L3Out. However, 'Advertise Externally' should still be selected. This is a less common use-case so it won't be discussed here.

The contract relationship between the L3Out and the EPG is required in order to cause the BD pervasive static route to get pushed to the BL. The actual route-advertisement is handled via redistribution of the static route into the external protocol. Lastly, the

redistribution route-maps will only be installed within the L3Outs that are associated to the BD. In this way the route isn't advertised out all L3Outs.

In this case, the BD subnet is 192.168.1.0/24 and it should be advertised via OSPF L3Out.

### Before applying the contract between the L3Out and internal EPG

```
leaf103# show ip route 192.168.1.0/24 vrf Prod:Vrf1
IP Route Table for VRF "Prod:Vrf1"
'*' denotes best ucast next-hop
***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
Route not found
```

Notice that the BD route isn't present yet on the BL.

### After applying the contract between the L3Out and internal EPG

At this point no other configuration has been done. The L3Out isn't yet associated to the BD and the 'Advertise Externally' flag isn't set.

```
leaf103# show ip route 10.0.1.0/24 vrf Prod:Vrf1
IP Route Table for VRF "Prod:Vrf1"
'*' denotes best ucast next-hop
***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
192.168.1.0/24, ubest/mbest: 1/0, attached, direct, pervasive
  *via 10.0.120.34%overlay-1, [1/0], 00:00:08, static, tag 4294967294
    recursive next hop: 10.0.120.34/32%overlay-1
```

Notice that the BD subnet route (indicated by the pervasive flag) is now deployed on the BL. Notice, however, that the route is tagged. This tag value is an implicit value assigned to BD routes before being configured with 'Advertise Externally'. All external protocols deny this tag from being redistributed.

### After selecting 'Advertise Externally' on the BD Subnet

The L3Out still hasn't been associated to the BD. However, notice that the tag has cleared.

```
leaf103# show ip route 192.168.1.0/24 vrf Prod:Vrf1
IP Route Table for VRF "Prod:Vrf1"
'*' denotes best ucast next-hop
 '**' denotes best mcast next-hop
 '[x/y]' denotes [preference/metric]
 '%<string>' in via output denotes VRF <string>
192.168.1.0/24, ubest/mbest: 1/0, attached, direct, pervasive
   *via 10.0.120.34%overlay-1, [1/0], 00:00:06, static
     recursive next hop: 10.0.120.34/32%overlay-1
```

At this point the route still isn't being advertised externally because there is no route-map and prefix-list that matches this prefix for redistribution into the external protocol. This can be verified with the following commands:

```
leaf103# show ip ospf vrf Prod:Vrf1
Routing Process default with ID 10.0.0.3 VRF Prod:Vrf1
Stateful High Availability enabled
Supports only single TOS(TOS0) routes
Supports opaque LSA
Table-map using route-map exp-ctx-2392068-deny-external-tag
Redistributing External Routes from
  static route-map exp-ctx-st-2392068
  direct route-map exp-ctx-st-2392068
  bgp route-map exp-ctx-PROTO-2392068
  eigrp route-map exp-ctx-PROTO-2392068
  coop route-map exp-ctx-st-2392068
```

The BD route is programmed as a static route, so check the static redistribution route-map by running 'show route-map <route-map name>' and then 'show ip prefix-list <name>' on any prefix-lists that are present in the route-map. Do this in the next step.

### After associating the L3Out to the BD

As mentioned earlier, this step results in the prefix-list that matches the BD subnet being installed in the static to external protocol redistribution route-map.

```
leaf103# show route-map exp-ctx-st-2392068
route-map exp-ctx-st-2392068, deny, sequence 1
Match clauses:
  tag: 4294967294
Set clauses:

...
route-map exp-ctx-st-2392068, permit, sequence 15803
Match clauses:
  ip address prefix-lists: IPv4-st16390-2392068-exc-int-inferred-export-dst
  ipv6 address prefix-lists: IPv6-deny-all
Set clauses:
  tag 0
```

Verify the prefix-list:

```
leaf103# show ip prefix-list IPv4-st16390-2392068-exc-int-inferred-export-dst
ip prefix-list IPv4-st16390-2392068-exc-int-inferred-export-dst: 1 entries
seq 1 permit 192.168.1.1/24
```

The BD subnet is being matched to redistribute into OSPF.

At this point the configuration and verification workflow is complete for the advertisement of the BD subnet out of the L3Out. Past this point, verification would be protocol specific. For instance:

- For EIGRP, verify that the route is being installed in the topology table with 'show ip eigrp topology vrf <name>'
- For OSPF, verify that the route is being installed in the database table as an External LSA with 'show ip ospf database vrf <name>'
- For BGP, verify that the route is in the BGP RIB with 'show bgp ipv4 unicast vrf <name>'

### BGP route advertisement

For BGP, all static routes are implicitly permitted for redistribution. The route-map which matches the BD subnet is applied at the BGP neighbor level.

```
leaf103# show bgp ipv4 unicast neighbor 10.0.0.134 vrf Prod:Vrf1 | grep Outbound
Outbound route-map configured is exp-13out-BGP-peer-2392068, handle obtained
```

In the above example, 10.0.0.134 is the BGP neighbor configured within the L3Out.

### EIGRP route advertisement

Like OSPF, a route-map is used to control Static to EIGRP redistribution. In this way only subnets associated to the L3Out and set to 'Advertise Externally' should be redistributed. This can be verified with this command:

```
leaf103# show ip eigrp vrf Prod:Vrf1
IP-EIGRP AS 100 ID 10.0.0.3 VRF Prod:Vrf1
Process-tag: default
Instance Number: 1
Status: running
Authentication mode: none
Authentication key-chain: none
Metric weights: K1=1 K2=0 K3=1 K4=0 K5=0
metric version: 32bit
IP proto: 88 Multicast group: 224.0.0.10
Int distance: 90 Ext distance: 170
Max paths: 8
Active Interval: 3 minute(s)
Number of EIGRP interfaces: 1 (0 loopbacks)
Number of EIGRP passive interfaces: 0
Number of EIGRP peers: 2
Redistributing:
  static route-map exp-ctx-st-2392068
  ospf-default route-map exp-ctx-PROTO-2392068
  direct route-map exp-ctx-st-2392068
  coop route-map exp-ctx-st-2392068
  bgp-65001 route-map exp-ctx-PROTO-2392068
```

The final working BD configuration is shown below.

### Bridge Domain L3 Configuration

The screenshot displays the Cisco APIC interface for configuring Bridge Domain L3. The left sidebar shows the navigation tree with 'Networking' > 'Bridge Domains' > 'BD1' > 'Subnets' selected. The main content area shows the configuration for Bridge Domain - BD1, with the 'Policy' and 'L3 Configurations' tabs highlighted. The 'Subnets' table shows the following configuration:

Gateway Address	Scope	Primary IP Address	Virtual IP	Subnet Control
192.168.1.1/24	Advertised Externally	False	False	

Below the table, the 'EP Move Detection Mode' is set to 'GARP based detection'. The 'Associated L3 Outs' section shows 'L3 Out' with 'OSPF' selected. The 'Show Usage', 'Reset', and 'Submit' buttons are visible at the bottom.

### Bridge domain route advertisement troubleshooting scenario

In this case, the typical symptom would normally be that a configured BD subnet is not being advertised out of an L3Out. Follow the previous workflow to understand which component is broken.

Start with the configuration before getting too low-level by verifying the following:

- Is there a contract between the EPG and L3Out?
- Is the L3Out associated to the BD?

- Is the BD subnet set to advertise externally?
- Is the external protocol adjacency up?

**Possible Cause: BD Not Deployed**

This case would be applicable in a couple of different scenarios, such as:

- The internal EPG is using VMM integration with On Demand option and no VM endpoints have been attached to the port-group for the EPG.
- The internal EPG has been created but no static path bindings have been configured or the interface on which the static path is configured are down.

In both cases, the BD would not be deployed and, as a result, the BD static route would not get pushed to the BL. The solution here is to deploy some active resources within an EPG which is linked to this BD so that the subnet gets deployed.

**Possible Cause: OSPF L3Out is configured as 'Stub' or 'NSSA' with No Redistribution**

When OSPF is used as the L3Out protocol, basic OSPF rules must still be followed. Stub areas do not allow redistributed LSAs but can advertise a default route instead. NSSA areas do allow redistributed paths but 'Send Redistributed LSAs into NSSA Area' must be selected on the L3Out. Or NSSA can also advertise a default route instead by disabling 'Originate Summary LSA' as well which is a typical scenario where 'Send Redistributed LSA's into NSSA Area' would be disabled.

**Possible Cause: 'Default-Export' Route-Profile with a 'Deny' Action configured under the L3Out**

When route-profiles are configured under an L3Out with the names of 'default-export' or 'default-import' they are implicitly applied to the L3Out. In addition, if the default-export route-profile is set to a deny action and configured as 'Match Prefix and Routing Policy' then BD subnets should be advertised out of this L3Out and would be implicitly denied:

## Default-export Deny Route Profile

The screenshot displays the Cisco APIC interface for configuring a Route Control Profile. The left-hand navigation pane shows the hierarchy: Tenants > Prod > L3Outs > OSPF > default-export. The main content area is titled 'Route Control Profile - default-export' and includes tabs for Policy, Faults, and History. The 'Policy' tab is active, showing the following configuration:

- Name: default-export
- Type: Match Prefix AND Routing Policy (selected) / Match Routing Policy Only
- Description: optional
- Contexts: A table with one entry:
 

Order	Name	Action	Description
0	deny1	Deny	

Buttons for 'Show Usage', 'Reset', and 'Submit' are located at the bottom right of the configuration area.

Prefix-matches within the default-export route-profile will not implicitly include BD Subnets if the 'Match Routing Policy Only' option is selected.

## External route import workflow

This section discusses how ACI learns external routes through an L3Out and distributes this to internal leaf nodes. It also covers transit and route-leaking use-cases in later sections

As with the previous section, the user should be aware of what happens at a higher-level.

By default, all routes learned through the external protocol are redistributed into the internal fabric BGP process. This is true regardless of what subnets are configured under the external EPG and what flags are selected. There are two examples where this is not true.

- If the 'Route Control Enforcement' option at the top level L3Out policy is set to 'Import'. In this case the route import model would go from a blacklist model (only specify what shouldn't be allowed) to a whitelist model (everything is implicitly denied unless configured otherwise).
- If the external Protocol is EIGRP or OSPF and an Interleak Route-Profile used does not match the external routes.

For an external route to be distributed to an internal leaf the following must happen:

- The route must be learned on the BL from the external router. To be a candidate to redistribute into the fabric MP-BGP process the route must be installed in the routing table rather than just in the protocol RIB.
- The route must be permitted to be redistributed or advertised into the internal BGP process. This should always happen unless import route-control enforcement or an Interleak Route-Profile is used.
- A BGP Route-Reflector Policy must be configured and applied to a Pod Policy Group which is applied to the Pod Profile. If this isn't applied, then the BGP Process will not initialize on the switches.

If the internal EPG/BD is in the same VRF as the L3Out then the above three steps are all that is required for the internal EPG/BD to use external routes.

### **Route is installed in BL routing table**

In this case the external route that should be learned on BLs 103 and 104 is 172.16.20.1/32.

```
leaf103# show ip route 172.16.20.1 vrf Prod:Vrf1
IP Route Table for VRF "Prod:Vrf1"
'*' denotes best ucast next-hop
 '**' denotes best mcast next-hop
 '[x/y]' denotes [preference/metric]
 '%<string>' in via output denotes VRF <string>

172.16.20.1/32, ubest/mbest: 1/0
   *via 10.10.34.3, vlan347, [110/20], 00:06:29, ospf-default, type-2
```

It is evident that it is installed in the routing table as being learned through OSPF. If it wasn't seen here, check the individual protocol and ensure adjacencies are up.

### Route is redistributed into BGP

The redistribution route-map can be verified, after checking that neither 'Import' enforcement or Interleak Route-Profiles are used, by looking at the route-map used for external protocol to BGP redistribution. See the following command:

```
leaf103# show bgp process vrf Prod:Vrf1

Information regarding configured VRFs:

BGP Information for VRF Prod:Vrf1
VRF Type                : System
VRF Id                  : 85
VRF state               : UP
VRF configured         : yes
VRF refcount           : 1
VRF VNID                : 2392068
Router-ID               : 10.0.0.3
Configured Router-ID   : 10.0.0.3
Confed-ID               : 0
Cluster-ID              : 0.0.0.0
MSITE Cluster-ID       : 0.0.0.0
No. of configured peers : 1
No. of pending config peers : 0
No. of established peers : 1
VRF RD                  : 101:2392068
VRF EVPN RD             : 101:2392068
...

Redistribution
  direct, route-map permit-all
  static, route-map imp-ctx-bgp-st-interleak-2392068
  ospf, route-map permit-all
  coop, route-map exp-ctx-st-2392068
  eigrp, route-map permit-all
```

Here it is evident that the 'permit-all' route-map is used for OSPF to BGP redistribution. This is the default. From here, BL can be verified and the local route originating from BGP checked:

```
a-leaf101# show bgp ipv4 unicast 172.16.20.1/32 vrf Prod:Vrf1
BGP routing table information for VRF Prod:Vrf1, address family IPv4 Unicast
BGP routing table entry for 172.16.20.1/32, version 25 dest ptr 0xa6f25ad0
Paths: (2 available, best #2)
Flags: (0x80c0002 00000000) on xmit-list, is not in urib, exported
  vpn: version 16316, (0x100002) on xmit-list
Multipath: eBGP iBGP

Advertised path-id 1, VPN AF advertised path-id 1
Path type: redistrib 0x408 0x1 ref 0 adv path ref 2, path is valid, is best path
AS-Path: NONE, path locally originated
0.0.0.0 (metric 0) from 0.0.0.0 (10.0.0.3)
  Origin incomplete, MED 20, localpref 100, weight 32768
  Extcommunity:
    RT:65001:2392068
    VNID:2392068
    COST:pre-bestpath:162:110

VRF advertise information:
Path-id 1 not advertised to any peer

VPN AF advertise information:
Path-id 1 advertised to peers:
10.0.64.64    10.0.72.66
Path-id 2 not advertised to any peer
```

In the above output, the 0.0.0.0/0 indicates it is originated locally. The list of peers advertised to are the spine nodes in the fabric which act as Route-Reflectors.

### Verify route on internal leaf

The BL should advertise it to the spine nodes through the VPNv4 BGP Address-Family. The spine nodes should advertise it to any leaf nodes with the VRF deployed (true of non-route-leaking example). On any of these leaf nodes run 'show bgp vpnv4 unicast <route> vrf overlay-1' to verify it is in VPNv4

Use the command below to verify the route on the internal leaf.

```
leaf101# show ip route 172.16.20.1 vrf Prod:Vrf1
IP Route Table for VRF "Prod:Vrf1"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.20.1/32, ubest/mbest: 2/0
 *via 10.0.72.64%overlay-1, [200/20], 00:21:24, bgp-65001, internal, tag 65001
   recursive next hop: 10.0.72.64/32%overlay-1
 *via 10.0.72.67%overlay-1, [200/20], 00:21:24, bgp-65001, internal, tag 65001
   recursive next hop: 10.0.72.67/32%overlay-1
```

In the above output the route is being learned through BGP and the next-hops should be the Physical TEPS (PTEP) of the BLs.

```
leaf101# acidiag fnvread
```

ID	Pod ID	Name	Serial Number	IP Address	Role	State	LastUpdMsgId
103	1	a-leaf101	FD020160TPS	10.0.72.67/32	leaf	active	0
104	1	a-leaf103	FD020160TQ0	10.0.72.64/32	leaf	active	0

### External route troubleshooting scenario

In this scenario the internal leaf (101) is not receiving an external route(s).

As always, first check the basics. Make sure that:

- Routing protocol adjacencies are up on the BLs.
- A BGP Route-Reflector Policy is applied to the Pod Policy-Group and the Pod Profile.

If the above criteria are correct, below are some more advanced examples of what could be causing the issue.

**Possible Cause: VRF not deployed on the internal leaf**

In this case, the issue would be that there are no EPGs with resources deployed on the internal leaf where the external route is expected. This could be caused by static path bindings only configured on down interfaces or only have On Demand mode VMM integrated EPGs present with no dynamic attachments detected.

Because the L3Out VRF is not deployed on the internal leaf (verify with 'show vrf' on internal leaf) the internal leaf will not import the BGP route from VPNv4.

To resolve this issue, the user should deploy resources within the L3Out VRF on the internal leaf.

**Possible Cause: Import Route Enforcement is being used**

As mentioned earlier, when import route-control enforcement is enabled the L3Out only accepts external routes that are explicitly permitted. Typically, the feature is implemented as a table-map. A table-map sits in between the protocol RIB and the actual routing table so that it only affects what is in the routing table.

In the output below the Import Route-Control is enabled, but there aren't any explicitly permitted routes. Notice that the LSA is in the OSPF database but not in the routing table on the BL:

```
leaf103# vsh -c "show ip ospf database external 172.16.20.1 vrf Prod:Vrf1"
      OSPF Router with ID (10.0.0.3) (Process ID default VRF Prod:Vrf1)

      Type-5 AS External Link States

Link ID          ADV Router      Age           Seq#           Checksum Tag
172.16.20.1     10.0.0.134     455          0x80000003    0xb9a0 0

leaf103# show ip route 172.16.20.1 vrf Prod:Vrf1
IP Route Table for VRF "Prod:Vrf1"
'*' denotes best ucast next-hop
 '**' denotes best mcast next-hop
 '[x/y]' denotes [preference/metric]
 '%<string>' in via output denotes VRF <string>

Route not found
```

Here is the table-map that is now installed causing this behavior:

```
leaf103# show ip ospf vrf Prod:Vrf1

Routing Process default with ID 10.0.0.3 VRF Prod:Vrf1
Stateful High Availability enabled
Supports only single TOS(TOS0) routes
Supports opaque LSA
Table-map using route-map exp-ctx-2392068-deny-external-tag
Redistributing External Routes from..

leaf103# show route-map exp-ctx-2392068-deny-external-tag
route-map exp-ctx-2392068-deny-external-tag, deny, sequence 1
  Match clauses:
    tag: 4294967295
  Set clauses:
route-map exp-ctx-2392068-deny-external-tag, deny, sequence 1999
Match clauses:
  ospf-area: 0.0.0.100
  Set clauses:
```

Anything learning in area 100, which is the area configured on this L3Out, is implicitly denied by this table-map so that it is not installed in the routing table.

To resolve this issue, the user should define the subnet on the external EPG with the 'Import Route Control Subnet' flag or create an Import Route-Profile that matches the prefixes to be installed.

- Note that import enforcement is not supported for EIGRP.
- Also note that for BGP, import enforcement is implemented as an inbound route-map applied to the BGP neighbor. Check the "BGP Route Advertisement" sub-section for details on how to check this.

#### **Possible Cause: an Interleak Profile is being used**

Interleak Route-Profiles are used for EIGRP and OSPF L3Outs and intended to allow for control over what is redistributed from the IGP into BGP as well as allows the application of policy such as setting BGP attributes.

Without an interleak Route-Profile, all routes are implicitly imported to BGP.

Without an interleak Route-Profile:

```
leaf103# show bgp process vrf Prod:Vrf1

Information regarding configured VRFs:

BGP Information for VRF Prod:Vrf1
VRF Type           : System
VRF Id            : 85
VRF state         : UP
VRF configured    : yes
VRF refcount      : 1
VRF VNID         : 2392068
Router-ID         : 10.0.0.3
Configured Router-ID : 10.0.0.3
Confed-ID         : 0
Cluster-ID        : 0.0.0.0
MSITE Cluster-ID  : 0.0.0.0
No. of configured peers : 1
No. of pending config peers : 0
No. of established peers : 1
VRF RD           : 101:2392068
VRF EVPN RD      : 101:2392068

...
Peers      Active-peers  Routes  Paths  Networks  Aggregates
1          1             7       11     0         0

Redistribution
  direct, route-map permit-all
  static, route-map imp-ctx-bgp-st-interleak-2392068
  ospf, route-map permit-all
  coop, route-map exp-ctx-st-2392068
  eigrp, route-map permit-all
```

With an interleak route-profile:

```
a-leaf103# show bgp process vrf Prod:Vrf1

Information regarding configured VRFs:

BGP Information for VRF Prod:Vrf1
VRF Type           : System
VRF Id            : 85
VRF state         : UP
VRF configured    : yes
VRF refcount      : 1
VRF VNID         : 2392068
Router-ID         : 10.0.0.3
Configured Router-ID : 10.0.0.3
```

```
Confed-ID           : 0
Cluster-ID          : 0.0.0.0
MSITE Cluster-ID    : 0.0.0.0
No. of configured peers : 1
No. of pending config peers : 0
No. of established peers : 1
VRF RD              : 101:2392068
VRF EVPN RD         : 101:2392068
```

```
...
```

```
Redistribution
  direct, route-map permit-all
  static, route-map imp-ctx-bgp-st-interleak-2392068
  ospf, route-map imp-ctx-proto-interleak-2392068
  coop, route-map exp-ctx-st-2392068
  eigrp, route-map permit-all
```

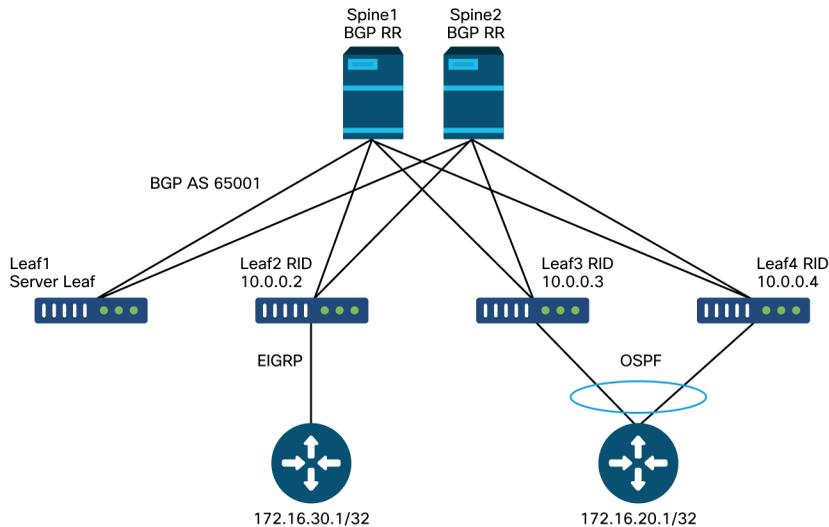
The above highlighted route-map would only permit what is explicitly matched in the configured Interleak Profile. If the external route isn't matched it will not be redistributed into BGP.

## Transit route advertisement workflow

This section discusses how routes from one L3Out are advertised out another L3Out. This would also cover the scenario where static routes that are configured directly on an L3Out need to be advertised. It will not go into every specific protocol consideration, but rather through how this is implemented in ACI. It will not go into inter-VRF transit routing at this time.

This scenario will use the following topology:

### Transit routing topology



The high-level flow of how 172.16.20.1 would be learned from OSPF and then advertised into EIGRP, and verifications of the whole process and troubleshooting scenarios, are discussed below.

For the 172.16.20.1 route to get advertised into EIGRP, one of the following must be configured:

- The subnet to be advertised could be defined on the EIGRP L3Out with the 'Export Route-Control Subnet' flag. As mentioned in the overview section, this flag is used mainly for transit routing and defines the subnets that should be advertised out of that L3Out.
- Configure 0.0.0.0/0 and select both 'Aggregate Export' and 'Export Route Control Subnet'. This creates a route-map for redistribution into the external

protocol that matches 0.0.0.0/0 and all prefixes that are more specific (which is an effective match any). Note that when 0.0.0.0/0 is used with 'Aggregate Export', static routes will not be matched for redistribution. This is to prevent inadvertently advertising BD routes that shouldn't be advertised.

- Lastly, it is possible to create an export route-profile that matches the prefixes to be advertised. Using this method could configure the 'Aggregate' option with prefixes besides 0.0.0.0/0.

The above configurations would result in the transit route being advertised but it still needs to have a security policy in place to allow dataplane traffic to flow. As with any EPG to EPG communication, a contract must be in place before traffic is permitted.

Note that duplicate external subnets with the 'External Subnet for External EPG' cannot be configured in the same VRF. When configured, subnets need to be more specific than 0.0.0.0. It is important to configure 'External Subnet for External EPG' only for the L3Out where the route is being received. Don't configure this on the L3Out that should be advertising this route.

It's also important to understand that all transit routes are tagged with a specific VRF Tag. By default, this tag is 4294967295. The Route-Tag policy is configured under 'Tenant > Networking > Protocols > Route-Tag':

## Route-Tag Policy

The screenshot shows the APIC interface for configuring a Route-Tag Policy. The left sidebar shows a tree view of configurations under 'Prod', with 'Route Tag' and its sub-item 'nonDefaultName' highlighted with a red box. The main panel displays the 'Protocol - Route Tag' configuration page, which contains a table with one entry: 'nonDefaultName' with a 'Tag' value of '11111' and an empty 'Description' field. The table has columns for Name, Tag, and Description. The interface also shows navigation tabs like System, Tenants, Fabric, etc., and a search bar at the top.

This Route Tag policy is then applied to the VRF. The purpose of this tag is essentially to prevent loops. This route tag is applied when the transit route is advertised back out of an L3Out. If these routes are then received back with the same route tag then the route is discarded.

### Verify that the route is present on the receiving BL via OSPF

Like the last section, first verify that the BL that should initially receive the correct route.

```
leaf103# show ip route 172.16.20.1 vrf Prod:Vrf1
IP Route Table for VRF "Prod:Vrf1"
'*' denotes best ucast next-hop
 '**' denotes best mcast next-hop
 '[x/y]' denotes [preference/metric]
 '%<string>' in via output denotes VRF <string>
```

```
172.16.20.1/32, ubest/mbest: 1/0
  *via 10.10.34.3, vlan347, [110/20], 01:25:30, ospf-default, type-2
```

For now, assume that the advertising L3Out is on a different BL (as in the topology) (later scenarios will discuss where it is on the same BL).

### Verify that the route is present in BGP on the receiving OSPF BL

For the OSPF route to be advertised to the external EIGRP router, the route needs to be advertised into BGP on the receiving OSPF BL

```
leaf103# show bgp ipv4 unicast 172.16.20.1/32 vrf Prod:Vrf1
BGP routing table information for VRF Prod:Vrf1, address family IPv4 Unicast
BGP routing table entry for 172.16.20.1/32, version 30 dest ptr 0xa6f25ad0
Paths: (2 available, best #1)
Flags: (0x80c0002 00000000) on xmit-list, is not in urib, exported
  vpn: version 17206, (0x100002) on xmit-list
Multipath: eBGP iBGP

Advertised path-id 1, VPN AF advertised path-id 1
Path type: redist 0x408 0x1 ref 0 adv path ref 2, path is valid, is best path
AS-Path: NONE, path locally originated
  0.0.0.0 (metric 0) from 0.0.0.0 (10.0.0.3)
    Origin incomplete, MED 20, localpref 100, weight 32768
    Extcommunity:
      RT:65001:2392068
      VNID:2392068
      COST:pre-bestpath:162:110

VRF advertise information:

Path-id 1 not advertised to any peer

VPN AF advertise information:
Path-id 1 advertised to peers:
  10.0.64.64      10.0.72.66
Path-id 2 not advertised to any peer
```

The route is in BGP.

## Verify on the EIGRP BL that should advertise the route that it is installed

```
leaf102# show ip route 172.16.20.1 vrf Prod:Vrf1
IP Route Table for VRF "Prod:Vrf1"
'*' denotes best ucast next-hop
 '**' denotes best mcast next-hop
 '[x/y]' denotes [preference/metric]
 '%<string>' in via output denotes VRF <string>

172.16.20.1/32, ubest/mbest: 2/0
 *via 10.0.72.67%overlay-1, [200/20], 00:56:46, bgp-65001, internal, tag 65001
   recursive next hop: 10.0.72.67/32%overlay-1
 *via 10.0.72.64%overlay-1, [200/20], 00:56:46, bgp-65001, internal, tag 65001
   recursive next hop: 10.0.72.64/32%overlay-1
```

It is installed in the routing table with overlay next-hops pointing to the originating border leaf nodes.

```
leaf102# acidiag fmvread
```

ID	Pod ID	Name	Serial Number	IP Address	Role	State	LastUpdMsgId
103	1	a-leaf101	FD020160TPS	<b>10.0.72.67/32</b>	leaf	active	0
104	1	a-leaf103	FD020160TQ0	<b>10.0.72.64/32</b>	leaf	active	0

**Verify that the route is advertised on the BL**

The route will be advertised by BL 102 as a result of the 'Export Route Control Subnet' flag being set on the configured subnet:

## Export Route Control

External EPG Instance Profile - instP

Policy Operational Stats Health Faults History

General Contracts Subject Labels EPG Labels

100

Properties

Configuration Status: applied

Configuration Issues:

Preferred Group Member: Exclude Include

Subnets:

IP Address	Scope	Name	Aggregate	Route Control Profile	Route Summarization Policy
0.0.0.0/0	External Subnets for the External EPG				
172.16.20.1/32	Export Route Control Subnet				

Show Usage Reset Submit

Current System Time: 2018-10-02T18:24:17C-04:00

Use the following command to view the route-map that is created as a result of this 'Export Route Control' flag:

```
leaf102# show ip eigrp vrf Prod:Vrf1
IP-EIGRP AS 101 ID 10.0.0.2 VRF Prod:Vrf1
  Process-tag: default
  Instance Number: 1
  Status: running
  Authentication mode: none
  Authentication key-chain: none
  Metric weights: K1=1 K2=0 K3=1 K4=0 K5=0
  metric version: 32bit
  IP proto: 88 Multicast group: 224.0.0.10
  Int distance: 90 Ext distance: 170
  Max paths: 8
  Active Interval: 3 minute(s)
  Number of EIGRP interfaces: 1 (0 loopbacks)
  Number of EIGRP passive interfaces: 0
  Number of EIGRP peers: 1
  Redistributing:
    static route-map exp-ctx-st-2392068
    ospf-default route-map exp-ctx-PROTO-2392068
    direct route-map exp-ctx-st-2392068
    coop route-map exp-ctx-st-2392068
bgp-65001 route-map exp-ctx-PROTO-2392068
```

To look for the 'BGP > EIGRP redistribution', look at the route-map. But, the route-map itself should be the same regardless of whether the source protocol is OSPF, EIGRP, or BGP. Static routes will be controlled with a different route-map.

```
leaf102# show route-map exp-ctx-PROTO-2392068
route-map exp-ctx-PROTO-2392068, permit, sequence 15801
  Match clauses:
    ip address prefix-lists: IPv4-PROTO32771-2392068-EXC-EXT-INFERRED-EXPORT-DST
    ipv6 address prefix-lists: IPv6-deny-all
  Set clauses:
    tag 4294967295

a-leaf102# show ip prefix-list IPv4-PROTO32771-2392068-EXC-EXT-INFERRED-EXPORT-DST
ip prefix-list IPv4-PROTO32771-2392068-EXC-EXT-INFERRED-EXPORT-DST: 1 entries
  seq 1 permit 172.16.20.1/32
```

In the above output, the VRF tag is set on this prefix for loop prevention and the subnet configured with 'Export Route Control' is explicitly matched.

### Transit Routing when receiving and advertising BL are the same

As discussed earlier, when the receiving and advertising BLs are different, the route must be advertised through the fabric using BGP. When the BLs are the same, the redistribution or advertisement can be done directly between the protocols on the leaf.

Below are brief descriptions of how this is implemented:

- **Transit routing between two OSPF L3Outs on the same leaf:** Route advertisement is controlled via an 'area-filter' applied to the OSPF process level. An L3Out in Area 0 must be deployed on the leaf since the routes are advertised between areas as opposed to through redistribution. Use 'show ip ospf vrf <name>' to view the filter-list. Display the contents of the filter using 'show route-map <filter name>'.  
`show ip ospf vrf <name>`  
`show route-map <filter name>`
- **Transit routing between OSPF and EIGRP L3Outs on the same leaf:** Route advertisement is controlled via redistribution route-maps that can be seen with 'show ip ospf' and 'show ip eigrp'. Note that if multiple OSPF L3Outs exist on the same BL the only way to redistribute into only one of those OSPF L3Outs is if the other is a Stub or NSSA with 'Send redistributed LSAs into NSSA area' disabled so that it doesn't allow any external LSA's.
- **Transit routing between OSPF or EIGRP and BGP on the same leaf:** Route advertisement into the IGP is controlled via redistribution route-maps. Route-advertisement into BGP is controlled via an outbound route-map applied directly to the bgp neighbor that the route should be sent to. This can be verified with 'show bgp ipv4 unicast neighbor <neighbor address> vrf <name> | grep Outbound'.  
`show bgp ipv4 unicast neighbor <neighbor address> vrf <name> | grep Outbound`
- **Transit routing between two BGP L3Outs on the same leaf:** All advertisement is controlled via route-maps applied directly to the bgp neighbor that the route should be sent to. This can be verified with 'show bgp ipv4 unicast neighbor <neighbor address> vrf <name> | grep Outbound'.  
`show bgp ipv4 unicast neighbor <neighbor address> vrf <name> | grep Outbound`

### Transit routing troubleshooting scenarios #1: Transit Route not advertised

This troubleshooting scenario involves routes that should be learned through one L3Out not being sent out the other L3Out.

As always, check the basics before looking at anything ACI specific.

- Are protocol adjacencies up?
- Is the route, that ACI should be advertising, learned from an external protocol in the first place?
- For BGP, is the path being dropped due to some BGP attribute? (as-path, etc.).
- Does the receiving L3Out have it in the OSPF database, EIGRP topology table, or BGP table?
- Is a BGP Route Reflector Policy applied to the Pod Policy Group that is applied to the Pod Profile?

If all the basic protocol verifications are configured correctly, below are some other common causes for a transit route that is not being advertised.

#### **Possible Cause: No OSPF Area 0**

If the affected topology involves two OSPF L3Outs on the same border leaf, then there must be an Area 0 for routes to be advertised from one area to another. Look at the "Transit routing between two OSPF L3Outs on the same leaf" bullet above for more details.

#### **Possible Cause: OSPF area is stub or NSSA**

This would be seen if the OSPF L3Out is configured with a Stub or NSSA area that is not configured to advertise external LSAs. With OSPF, external LSAs are never advertised into Stub areas. They are advertised into NSSA areas if 'Send Redistributed LSAs into NSSA Area' is selected.

### Transit routing troubleshooting scenarios #2: Transit Route not received

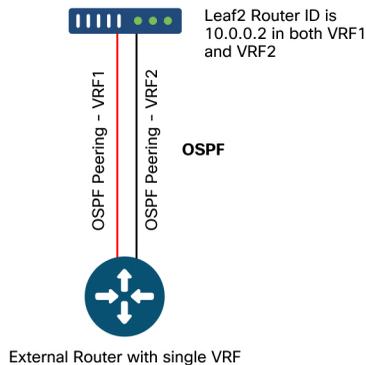
In this scenario the problem is that some routes advertised by an ACI L3Out are not being received back in another L3Out. This scenario could be applicable if the L3Outs are in two separate fabrics and are connected by external routers or if the L3Outs are in different VRFs and the routes are being passed between the VRFs by an external router.

#### Possible Cause: BL is Configured with the same Router ID in multiple VRFs

From a configuration perspective, a router-id cannot be duplicated within the same VRF. However, it is typically fine to use the same router-id in different VRFs as long as the two VRFs aren't attached to the same routing protocol domains.

Consider the following topology:

#### External router with single VRF – Transit Route not received



The problem here would be that the ACI leaf sees LSAs with its own Router-ID being received, resulting in these not being installed in the OSPF database.

In addition, if the same setup was seen with VPC pairs, LSAs would continuously be added and deleted on some routers. For example, the router would see LSAs coming

from its VPC peer with VRF and LSAs coming from the same node (with same Router-ID) that were originated in the other VRF.

To resolve this issue, the user should make sure that a node will have a different, unique router-id within each VRF that it has an L3Out in.

**Possible cause: routes from one L3Out in one ACI fabric received on another fabric with same VRF tag**

The default route-tag in ACI is always the same unless it is changed. If routes are advertised from one L3Out in one VRF or ACI fabric to another L3Out in another VRF or ACI fabric without changing the default VRF tags, the routes will be dropped by the receiving BLs.

The solution to this scenario is simply to use a unique Route-Tag policy for each VRF in ACI.

### Transit routing troubleshooting scenarios #3 – Transit Routes unexpectedly advertised

This scenario would be seen when transit routes are advertised out an L3Out where they are not intended to be advertised.

**Possible cause: usage of 0.0.0.0/0 with 'Aggregate Export'**

When an external subnet is configured as 0.0.0.0/0 with 'Export Route Control Subnet' and 'Aggregate Export' the result is that a match all redistribution route-map is installed. In this case all routes on the BL that were learned through OSPF, EIGRP, or BGP are advertised out the L3Out where this is configured.

Below is the route-map that is deployed to the leaf as a result of the Aggregate Export:

```
leaf102# show ip eigrp vrf Prod:Vrf1
IP-EIGRP AS 101 ID 10.0.0.2 VRF Prod:Vrf1
Process-tag: default
Instance Number: 1
Status: running
Authentication mode: none
Authentication key-chain: none
Metric weights: K1=1 K2=0 K3=1 K4=0 K5=0
metric version: 32bit
IP proto: 88 Multicast group: 224.0.0.10
```

```

Int distance: 90 Ext distance: 170
Max paths: 8
Active Interval: 3 minute(s)
Number of EIGRP interfaces: 1 (0 loopbacks)
Number of EIGRP passive interfaces: 0
Number of EIGRP peers: 1
Redistributing:
  static route-map exp-ctx-st-2392068
  ospf-default route-map exp-ctx-PROTO-2392068
  direct route-map exp-ctx-st-2392068
  coop route-map exp-ctx-st-2392068
  bgp-65001 route-map exp-ctx-PROTO-2392068
Tablemap: route-map exp-ctx-2392068-deny-external-tag , filter-configured
Graceful-Restart: Enabled
Stub-Routing: Disabled
NSF converge time limit/expiries: 120/0
NSF route-hold time limit/expiries: 240/0
NSF signal time limit/expiries: 20/0
Redistributed max-prefix: Disabled
selfAdvRtTag: 4294967295
leaf102# show route-map exp-ctx-PROTO-2392068
route-map exp-ctx-PROTO-2392068, permit, sequence 19801
Match clauses:
  ip address prefix-lists: IPv4-PROTO32771-2392068-agg-ext-inferred-export-dst
  ipv6 address prefix-lists: IPv6-deny-all
Set clauses:
  tag 4294967295

leaf102# show ip prefix-list IPv4-PROTO32771-2392068-agg-ext-inferred-export-dst
  ip prefix-list IPv4-PROTO32771-2392068-agg-ext-inferred-export-dst: 1 entries
seq 1 permit 0.0.0.0/0 le 32

```

This is the number one cause of routing loops that involve an ACI environment.

# Contract and L3Out

## Prefix-based EPG on L3Out

In an internal EPG (non-L3Out), contracts are enforced after deriving the pcTag of the source and the pcTag of the destination EPG. The encapsulation VLAN/VXLAN of the packet received on the downlink port is used to drive this pcTag by classing the packet into the EPG. Whenever learning a MAC address or an IP address, it is learned along with its access encapsulation and the associated EPG pcTag. For more details on pcTag and contract enforcement, please refer to the "Security policies" chapter.

L3Outs also drive a pcTag using its L3Out EPG (External EPG) located under 'Tenant > Networking > L3OUT > Networks > L3OUT-EPG'. However, L3Outs do not rely on VLANs and interfaces to classify packets as such. Classification is instead based on source prefix/subnet in a 'Longest Prefix Match' fashion. Hence, an L3Out EPG can be referred to as a **prefix-based EPG**. After a packet is classified into an L3Out based on a subnet, it follows a similar policy enforcement pattern as a regular EPG.

The following diagram outlines where the pcTag of a given L3Out EPG can be found within the GUI.

### Location of the pcTag for an L3Out

The screenshot displays the Cisco APIC interface for configuring an External EPG Instance Profile. The left-hand navigation pane shows the hierarchy: Prod > External Bridged Networks > L3Outs > External EPGs > L3OUT-EPG. The main configuration area is titled 'External EPG Instance Profile - L3OUT-EPG'. Under the 'Properties' section, the 'pcTag' field is highlighted with a red box and contains the value '32772'. Other visible fields include Name (L3OUT-EPG), Alias, Tags, Global Alias, Description (optional), Contract Exception Tag, Configured VRF Name (VRF1), Resolved VRF (uni/tn-Prod/ctx-VRF1), QoS Class (Unspecified), and Target DSCP (Unspecified).

The user is responsible for defining the prefix-based EPG table. This is done using the 'External Subnet for External EPG' subnet scope. Each subnet set with that scope will add an entry in a static Longest Prefix Match (LPM) table. This subnet will point to the pcTag value that will get used for any IP address falling within that prefix.

The LPM table of prefix-based EPG subnets can be verified on leaf switches using the following command:

```
vsh -c 'show system internal policy-mgr prefix'
```

## Remarks:

- LPM table entries are scoped to VRF VNID. The lookup is done per vrf\_vnid/src pcTag/dst pcTag.
- Each entry points to a single pcTag. As a consequence, two L3Out EPGs cannot use the same subnet with the same mask length within the same VRF.
- Subnet 0.0.0.0/0 always uses special pcTag 15. As such, it can be duplicated but should only be done so with a full understanding of the policy enforcement implications.
- This table is used in both directions.
  - From L3Out to Leaf Local Endpoint, the source pcTag is derived using this table.
  - From Leaf Local Endpoint to L3Out, the destination pcTag is derived using this table.
- If the VRF has the 'Ingress' enforcement setting for 'Policy Control Enforcement Direction', then the LPM prefix table will be present on the L3Out BLs as well as any leaf switches in the VRF that have a contract with the L3Out.

## Example 1: Single L3Out with specific prefix

Scenario: A single BGP L3Out in vrf Prod:VRF1 with one L3Out EPG. Prefix 172.16.1.0/24 is being received from an external source so it must be classified into the L3Out EPG.

```
bdsol-aci32-leaf3# show ip route 172.16.1.0 vrf Prod:VRF1
IP Route Table for VRF "Prod:VRF1"
'*' denotes best ucast next-hop
'***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

172.16.1.0/24, ubest/mbest: 1/0
  *via 10.0.0.134%Prod:VRF1, [20/0], 00:56:14, bgp-132, external, tag 65002
    recursive next hop: 10.0.0.134/32%Prod:VRF1
```

First, add the subnet to the prefix table.

Subnet with 'External Subnets for the External EPG' scope

### Create Subnet

IP Address:   
address/mask

Name:

scope:  Export Route Control Subnet  
 Import Route Control Subnet  
 External Subnets for the External EPG  
 Shared Route Control Subnet  
 Shared Security Import Subnet

BGP Route Summarization Policy:

aggregate:  Aggregate Export  
 Aggregate Import  
 Aggregate Shared Routes

Route Control Profile:

Name	Direction
------	-----------

Verify the programming of the prefix list on the leaf switches that have the VRF of the L3Out:

```

bdso1-aci32-leaf3# vsh -c ' show system internal policy-mgr prefix ' | egrep "Prod|==|Addr"
Vrf-Vni VRF-Id Table-Id Table-State VRF-Name Addr Class Shared Remote Complete
-----
2097154 35 0x23 Up Prod:VRF1 0.0.0.0/0 15 True True False
2097154 35 0x23 Up Prod:VRF1 172.16.1.0/24 32772 True True False

```

The pcTag of the L3Out EPG is 32772 in vrf scope 2097154.

## Example 2: Single L3Out with multiple prefixes

Expanding on the previous example, in this scenario the L3Out is receiving multiple prefixes. While entering each prefix is functionally sound, an alternative option (depending on the intended design) is to accept all prefixes received on the L3Out.

This can be accomplished with the '0.0.0.0/0' prefix.

[default subnet with 'External Subnets for the External EPG' scope](#)

Subnet - 0.0.0.0/0

The screenshot shows the configuration page for a subnet with IP address 0.0.0.0/0. The 'Policy' tab is selected. Under the 'Scope' section, the checkbox for 'External Subnets for the External EPG' is checked. Other options like 'Export Route Control Subnet', 'Import Route Control Subnet', 'Shared Route Control Subnet', and 'Shared Security Import Subnet' are unchecked. Under the 'Aggregate' section, 'Aggregate Export', 'Aggregate Import', and 'Aggregate Shared Routes' are also unchecked. The 'BGP Route Summarization Policy' is set to 'select an option'. The 'Route Control Profile' section is currently empty, displaying the message: 'No items have been found. Select Actions to create a new item.'

This results in the following policy-mgr prefix table entry:

```
bdsol-aci32-leaf3# vsh -c ' show system internal policy-mgr prefix ' | egrep "Prod|==|Addr"
Vrf-Vni VRF-Id Table-Id Table-State VRF-Name Addr Class Shared Remote Complete
-----
2097154 35 0x23 Up Prod:VRF1 0.0.0.0/0 15 True True False
2097154 35 0x23 Up Prod:VRF1 172.16.1.0/24 32772 True True False
```

Note that the pcTag assigned to 0.0.0.0/0 uses value 15, not 32772. pcTag 15 is a reserved system pcTag which is only used with 0.0.0.0/0 which acts as wildcard to match all prefixes on an L3Out.

If the VRF has a single L3Out with a single L3Out EPG using the 0.0.0.0/0, then the policy-prefix remains unique and is the easiest approach to catch all.

### Example 3a: Multiple L3Out EPGs in a VRF

In this scenario there are multiple L3Out EPGs in the same VRF.

Note: From a prefix-based EPG perspective, the following two configurations will result in equivalent LPM policy-mgr prefix table entries:

- 1 Two L3Outs with one L3Out EPG each.
- 2 One L3Out with two L3Out EPGs

In both scenarios, the total number of L3Out EPGs is 2. This means that each one will have its own pcTag and associated subnets.

All pcTags of a given L3Out EPG can be viewed in the GUI at 'Tenant > Operational > Resource id > L3Outs'

## Verification of the L3Out pcTag

The screenshot shows the Cisco ACI GUI for a Tenant named 'Prod'. The 'Operational' tab is selected, and the 'L3Outs' section is highlighted. A table lists two L3Outs: 'L3OUT-EPG' and 'L3OUT-EPG2', both with a Class ID and a Scope.

EPG Name	EPG Alias	Class ID	Scope
L3OUT-EPG		32772	2097154
L3OUT-EPG2		32773	2097154

In this scenario, the ACI fabric is receiving multiple prefixes from the external routers and the L3Out EPG definition is as follows:

- 172.16.1.0/24 assigned to L3OUT-EPG.
- 172.16.2.0/24 assigned to L3OUT-EPG2.
- 172.16.0.0/16 assigned to L3OUT-EPG (to catch the 172.16.3.0/24 prefix).

To match this, the config will be defined as follows:

- L3OUT-EPG has subnet 172.16.1.0/24 and 172.16.0.0/16 both with scope 'External Subnet for the External EPG'.
- L3OUT-EPG2 has subnet 172.16.2.0/24 with scope 'External Subnet for the External EPG'.

The resulting prefix table entries will be:

```

bdso1-aci32-leaf3# vsh -c 'show system internal policy-mgr prefix' | egrep "Prod|==|Addr"
Vrf-Vni VRF-Id Table-Id Table-State VRF-Name Addr Class Shared Remote Complete
=====
2097154 35 0x23 Up Prod:VRF1 0.0.0.0/0 15 True True False
2097154 35 0x23 Up Prod:VRF1 172.16.1.0/24 32772 True True False
2097154 35 0x23 Up Prod:VRF1 172.16.0.0/16 32772 True True False
2097154 35 0x23 Up Prod:VRF1 172.16.2.0/24 32773 True True False

```

172.16.2.0/24 is assigned to pcTag 32773 (L3OUT-EPG2) and 172.16.0.0/16 is assigned to 32772 (L3OUT-EPG).

In this scenario, the entry for 172.16.1.0/24 is redundant as the /16 supernet is assigned to the same EPG.

Multiple L3Out EPGs is useful when the goal is to apply different contracts to groups of prefixes within a single L3Out. The next example will illustrate how contracts come into play with multiple L3Out EPGs.

### Example 3b: multiple L3Out EPGs with different contracts

This scenario contains the following setup:

- ICMP contract allowing only ICMP.
- HTTP contract allowing only tcp destination port 80.
- EPG1 (pcTag 32770) provides the HTTP contract consumed by L3OUT-EPG (pcTag 32772).
- EPG2 (pcTag 32771) provides the ICMP contract consumed by L3OUT-EPG2 (pcTag 32773).

The same policymgr prefixes from the previous example will be used:

- 172.16.1.0/24 in L3OUT-EPG should permit HTTP to EPG1
- 172.16.2.0/24 in L3OUT-EPG2 should permit ICMP to EPG2

policy-mgr prefix and zoning-rules:

```

bdsol-aci32-leaf3# vsh -c ' show system internal policy-mgr prefix ' | egrep "Prod|==|Addr"
Vrf-Vni Vrf-Id Table-Id Table-State VRF-Name Addr Class Shared Remote Complete
=====
2097154 35 0x23 Up Prod:VRF1 0.0.0.0/0 15 True True False
2097154 35 0x23 Up Prod:VRF1 172.16.1.0/24 32772 True True False
2097154 35 0x23 Up Prod:VRF1 172.16.0.0/16 32772 True True False
2097154 35 0x23 Up Prod:VRF1 172.16.2.0/24 32773 True True False

```

```

bdsol-aci32-leaf3# show zoning-rule scope 2097154
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir | operSt | Scope | Name | Action | Priority |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 4326 | 0 | 0 | implicit | uni-dir | enabled | 2097154 | | deny_log | any_any_any(21) |
| 4335 | 0 | 16387 | implicit | uni-dir | enabled | 2097154 | | permit | any_dest_any(16) |
| 4334 | 0 | 0 | implarp | uni-dir | enabled | 2097154 | | permit | any_any_filter(17) |
| 4333 | 0 | 15 | implicit | uni-dir | enabled | 2097154 | | deny_log | any_vrf_any_deny(22) |
| 4332 | 0 | 16386 | implicit | uni-dir | enabled | 2097154 | | permit | any_dest_any(16) |
| 4342 | 32771 | 32773 | 5 | uni-dir-ignore | enabled | 2097154 | ICMP | permit | fully_qual(7) |
| 4343 | 32773 | 32771 | 5 | bi-dir | enabled | 2097154 | ICMP | permit | fully_qual(7) |
| 4340 | 32770 | 32772 | 38 | uni-dir | enabled | 2097154 | HTTP | permit | fully_qual(7) |
| 4338 | 32772 | 32770 | 37 | uni-dir | enabled | 2097154 | HTTP | permit | fully_qual(7) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

## Datapath validation using fTriage – flow allowed by policy

With an ICMP flow between 172.16.2.1 on the external network and 192.168.3.1 in EPG2, fTriage can be used to catch and analyze the flow. In this case, start fTriage on both leaf switch 103 and 104 as traffic may enter either of them:

```

admin@apic1:~> ftriage route -ii LEAF:103,104 -sip 172.16.2.1 -dip 192.168.3.1
fTriage Status: {"dbgftriage": {"attributes": {"operState": "InProgress", "pid": "14454", "apicId": "1", "id": "0"}}}
Starting ftriage
Log file name for the current run is: ftlog_2019-10-02-22-30-41-871.txt
2019-10-02 22:30:41,874 INFO /controller/bin/ftriage route -ii LEAF:103,104 -sip 172.16.2.1 -dip 192.168.3.1
2019-10-02 22:31:28,868 INFO ftriage: main:1165 Invoking ftriage with default password and default
username: apic#fallback\admin
2019-10-02 22:32:15,076 INFO ftriage: main:839 L3 packet Seen on bdsol-aci32-leaf3 Ingress: Eth1/12
(Po1) Egress: Eth1/12 (Po1) Vnid: 11365
2019-10-02 22:32:15,295 INFO ftriage: main:242 ingress encap string vlan-2551
2019-10-02 22:32:17,839 INFO ftriage: main:271 Building ingress BD(s), Ctx
2019-10-02 22:32:20,583 INFO ftriage: main:294 Ingress BD(s) Prod:VRF1:l3out-BGP:vlan-2551
2019-10-02 22:32:20,584 INFO ftriage: main:301 Ingress Ctx: Prod:VRF1

```

```

2019-10-02 22:32:20,693 INFO ftrriage: pktrec:490 bdsol-aci32-leaf3: Collecting transient losses snapshot
for LC module: 1
2019-10-02 22:32:38,933 INFO ftrriage: nxos:1404 bdsol-aci32-leaf3: nxos matching rule id:4343 scope:34
filter:5
2019-10-02 22:32:39,931 INFO ftrriage: main:522 Computed egress encap string vlan-2502
2019-10-02 22:32:39,933 INFO ftrriage: main:313 Building egress BD(s), Ctx
2019-10-02 22:32:41,796 INFO ftrriage: main:331 Egress Ctx Prod:VRF1
2019-10-02 22:32:41,796 INFO ftrriage: main:332 Egress BD(s): Prod:BD2
2019-10-02 22:32:48,636 INFO ftrriage: main:933 SIP 172.16.2.1 DIP 192.168.3.1
2019-10-02 22:32:48,637 INFO ftrriage: unicast:973 bdsol-aci32-leaf3: <- is ingress node
2019-10-02 22:32:51,257 INFO ftrriage: unicast:1202 bdsol-aci32-leaf3: Dst EP is local
2019-10-02 22:32:54,129 INFO ftrriage: misc:657 bdsol-aci32-leaf3: EP if(Po1) same as egr if(Po1)
2019-10-02 22:32:55,348 INFO ftrriage: misc:657 bdsol-aci32-leaf3: DMAC(00:22:BD:F8:19:FF) same as
RMAC(00:22:BD:F8:19:FF)
2019-10-02 22:32:55,349 INFO ftrriage: misc:659 bdsol-aci32-leaf3: L3 packet getting routed/bounced in
SUG
2019-10-02 22:32:55,596 INFO ftrriage: misc:657 bdsol-aci32-leaf3: Dst IP is present in SUG L3 tbl
2019-10-02 22:32:55,896 INFO ftrriage: misc:657 bdsol-aci32-leaf3: RW segid:11365 in SUG same as EP
segid:11365
2019-10-02 22:33:02,150 INFO ftrriage: main:961 Packet is Exiting fabric with peer-device: bdsol-aci32-
n3k-3 and peer-port: Ethernet1/16

```

fTriage confirms the zoning-rule hit against the ICMP rule from L3OUT\_EPG2 to EPG:

```

2019-10-02 22:32:38,933 INFO ftrriage: nxos:1404 bdsol-aci32-leaf3: nxos matching rule id:4343 scope:34
filter:5

```

## Datapath validation using fTriage – flow that is not allowed by policy

With ICMP traffic sourced from 172.16.1.1 (L3OUT-EPG) towards 192.168.3.1 (EPG2), expect a policy drop.

```

admin@apic1:~> ftrriage route -ii LEAF:103,104 -sip 172.16.1.1 -dip 192.168.3.1
fTriage Status: {"dbgFtrriage": {"attributes": {"operState": "InProgress", "pid": "15139", "apicId": "1", "id":
"0"}}}
Starting ftrriage
Log file name for the current run is: ftlog_2019-10-02-22-39-15-050.txt
2019-10-02 22:39:15,056 INFO /controller/bin/ftrriage route -ii LEAF:103,104 -sip 172.16.1.1 -dip 192.168.3.1
2019-10-02 22:40:03,523 INFO ftrriage: main:1165 Invoking ftrriage with default password and default
username: apic#fallback\admin
2019-10-02 22:40:43,338 ERROR ftrriage: unicast:234 bdsol-aci32-leaf3: L3 packet getting fwd dropped,
checking drop reason
2019-10-02 22:40:43,339 ERROR ftrriage: unicast:234 bdsol-aci32-leaf3: L3 packet getting fwd dropped,
checking drop reason
SECURITY_GROUP_DENY condition setcast:236 bdsol-aci32-leaf3: Drop reason - SECURITY_GROUP_DENY
condition set
2019-10-02 22:40:43,340 INFO ftrriage: unicast:252 bdsol-aci32-leaf3: policy drop flow sclass:32772
dclass:32771 sg_label:34 proto:3

```

```
2019-10-02 22:40:43,340 INFO fTriage: main:681 : FTriage Completed with hunch: None
fTriage Status: {"dbgfTriage": {"attributes": {"operState": "Idle", "pid": "0", "apicId": "0", "id": "0"}}}
```

fTriage confirms that the packet is dropped with the SECURITY\_GROUP\_DENY (policy drop) reason and that the derived source pcTag is 32772 and destination pcTag is 32771. Checking this against zoning-rules, there are clearly no entries between those EPG.

```
bdsol-aci32-leaf3# show zoning-rule scope 2097154 src-epg 32772 dst-epg 32771
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir | operSt | Scope | Name | Action | Priority |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

## Example 4: multiple L3Outs with multiple prefixes

The scenario is setup similarly to example 3 (L3Out and L3Out EPG definitions), but the network defined on both L3Out EPGs is 0.0.0.0/0.

Contract configuration is the following:

- ICMP1 contract allowing ICMP.
- ICMP2 contract allowing ICMP.
- EPG1 (pcTag 32770) provides ICMP1 contract which is consumed by L3OUT-EPG (pcTag 32772).
- EPG2 (pcTag 32771) provides ICMP2 contract which is consumed by L3OUT-EPG2 (pcTag 32773).

This configuration may look ideal in the case where the external network is advertising many prefixes, but there are at least two chunks of prefixes that follow different allowed flow patterns. In this example, one prefix should only allow ICMP1 and the other should only allow ICMP2.

Despite using '0.0.0.0/0' twice in the same VRF, only one prefix gets programmed in the policy-mgr prefix table:

```

bdsol-aci32-leaf3# vsh -c ' show system internal policy-mgr prefix ' | egrep "Prod|==|Addr"
Vrf-Vni Vrf-Id Table-Id Table-State VRF-Name Addr Class
Shared Remote Complete
=====
=====
=====
2007154 35 0x23 Up Prod:VRF1 0.0.0.0/0 15
True True False

```

Two flows reexamined below. Based on the contract configuration above, the following is expected:

- 1 172.16.2.1 (L3OUT-EPG2) to 192.168.3.1 (EPG2) **should** be allowed by ICMP2
- 2 172.16.2.1 (L3OUT-EPG2) to 192.168.1.1 (EPG1) **should not** be allowed as there is no contract between EPG1 and L3OUT-EPG2

### Datapath validation using fTriage – flow that is allowed by policy

Run fTriage with an ICMP flow from 172.16.2.1 (L3OUT-EPG2) to 192.168.3.1 (EPG2 – pCtag 32771).

```

admin@apic1:~> ftrriage route -ii LEAF:103,104 -sip 172.16.2.1 -dip 192.168.3.1
fTriage Status: {"dbgFtrriage": {"attributes": {"operState": "InProgress", "pid": "1921", "apicId": "1", "id": "0"}}}
Starting ftrriage
Log file name for the current run is: ftlog_2019-10-02-23-11-14-298.txt
2019-10-02 23:11:14,302 INFO /controller/bin/ftrriage route -ii LEAF:103,104 -sip 172.16.2.1 -dip 192.168.3.1
2019-10-02 23:12:00,887 INFO ftrriage: main:1165 Invoking ftrriage with default password and default
username: apic#fallback\admin
2019-10-02 23:12:44,565 INFO ftrriage: main:839 L3 packet Seen on bdsol-aci32-leaf3 Ingress: Eth1/12
(Po1) Egress: Eth1/12 (Po1) Vnid: 11365
2019-10-02 23:12:44,782 INFO ftrriage: main:242 ingress encap string vlan-2551
2019-10-02 23:12:47,260 INFO ftrriage: main:271 Building ingress BD(s), Ctx
2019-10-02 23:12:50,041 INFO ftrriage: main:294 Ingress BD(s) Prod:VRF1:L3out-BGP:vlan-2551
2019-10-02 23:12:50,042 INFO ftrriage: main:301 Ingress Ctx: Prod:VRF1
2019-10-02 23:12:50,151 INFO ftrriage: pktrec:490 bdsol-aci32-leaf3: Collecting transient losses snapshot
for LC module: 1
2019-10-02 23:13:08,595 INFO ftrriage: nxos:1404 bdsol-aci32-leaf3: nxos matching rule id:4336 scope:34
filter:5
2019-10-02 23:13:09,608 INFO ftrriage: main:522 Computed egress encap string vlan-2502

```

```

2019-10-02 23:13:09,609 INFO ftriage: main:313 Building egress BD(s), Ctx
2019-10-02 23:13:11,449 INFO ftriage: main:331 Egress Ctx Prod:VRF1
2019-10-02 23:13:11,449 INFO ftriage: main:332 Egress BD(s): Prod:BD2
2019-10-02 23:13:18,383 INFO ftriage: main:933 SIP 172.16.2.1 DIP 192.168.3.1
2019-10-02 23:13:18,384 INFO ftriage: unicast:973 bdsol-aci32-leaf3: <- is ingress node
2019-10-02 23:13:21,078 INFO ftriage: unicast:1202 bdsol-aci32-leaf3: Dst EP is local
2019-10-02 23:13:23,926 INFO ftriage: misc:657 bdsol-aci32-leaf3: EP if(Po1) same as egr if(Po1)
2019-10-02 23:13:25,216 INFO ftriage: misc:657 bdsol-aci32-leaf3: DMAC(00:22:BD:F8:19:FF) same as
RMAC(00:22:BD:F8:19:FF)
2019-10-02 23:13:25,217 INFO ftriage: misc:659 bdsol-aci32-leaf3: L3 packet getting routed/bounced in
SUG
2019-10-02 23:13:25,465 INFO ftriage: misc:657 bdsol-aci32-leaf3: Dst IP is present in SUG L3 tbl
2019-10-02 23:13:25,757 INFO ftriage: misc:657 bdsol-aci32-leaf3: RW seg_id:11365 in SUG same as EP
segid:11365
2019-10-02 23:13:32,235 INFO ftriage: main:961 Packet is Exiting fabric with peer-device: bdsol-aci32-
n3k-3 and peer-port: Ethernet1/16

```

This flow is allowed (as expected) by zoning-rule 4336.

### Datapath validation using fTriage – flow that is not allowed by policy

Run fTriage with an ICMP flow from 172.16.2.1 (L3OUT-EPG2) to 192.168.1.1 (EPG1 – pcTag 32770):

```

admin@apic1:> ftriage route -ii LEAF:103,104 -sip 172.16.2.1 -dip 192.168.1.1
fTriage Status: {"dbgftriage": {"attributes": {"operState": "InProgress", "pid": "31500", "apicId": "1", "id":
"0"}}}
Starting ftriage
Log file name for the current run is: ftlog_2019-10-02-23-53-03-478.txt
2019-10-02 23:53:03,482 INFO /controller/bin/ftriage route -ii LEAF:103,104 -sip 172.16.2.1 -dip 192.168.1.1
2019-10-02 23:53:50,014 INFO ftriage: main:1165 Invoking ftriage with default password and default
username: apic#fallback\admin
2019-10-02 23:54:39,199 INFO ftriage: main:839 L3 packet Seen on bdsol-aci32-leaf3 Ingress: Eth1/12
(Po1) Egress: Eth1/12 (Po1) Vnid: 11364
2019-10-02 23:54:39,417 INFO ftriage: main:242 ingress encap string vlan-2551
2019-10-02 23:54:41,962 INFO ftriage: main:271 Building ingress BD(s), Ctx
2019-10-02 23:54:44,765 INFO ftriage: main:294 Ingress BD(s) Prod:VRF1:l3out-BGP:vlan-2551
2019-10-02 23:54:44,766 INFO ftriage: main:301 Ingress Ctx: Prod:VRF1
2019-10-02 23:54:44,875 INFO ftriage: pktrec:490 bdsol-aci32-leaf3: Collecting transient losses snapshot
for LC module: 1
2019-10-02 23:55:02,905 INFO ftriage: nxos:1404 bdsol-aci32-leaf3: nxos matching rule id:4341 scope:34
filter:5
2019-10-02 23:55:04,525 INFO ftriage: main:522 Computed egress encap string vlan-2501
2019-10-02 23:55:04,526 INFO ftriage: main:313 Building egress BD(s), Ctx
2019-10-02 23:55:06,390 INFO ftriage: main:331 Egress Ctx Prod:VRF1
2019-10-02 23:55:06,390 INFO ftriage: main:332 Egress BD(s): Prod:BD1
2019-10-02 23:55:13,571 INFO ftriage: main:933 SIP 172.16.2.1 DIP 192.168.1.1
2019-10-02 23:55:13,572 INFO ftriage: unicast:973 bdsol-aci32-leaf3: <- is ingress node
2019-10-02 23:55:16,159 INFO ftriage: unicast:1202 bdsol-aci32-leaf3: Dst EP is local

```

```

2019-10-02 23:55:18,949 INFO ftrriage: misc:657 bdsol-aci32-leaf3: EP if(Po1) same as egr if(Po1)
2019-10-02 23:55:20,126 INFO ftrriage: misc:657 bdsol-aci32-leaf3: DMAC(00:22:BD:F8:19:FF) same as
RMAC(00:22:BD:F8:19:FF)
2019-10-02 23:55:20,126 INFO ftrriage: misc:659 bdsol-aci32-leaf3: L3 packet getting routed/bounced in
SUG
2019-10-02 23:55:20,395 INFO ftrriage: misc:657 bdsol-aci32-leaf3: Dst IP is present in SUG L3 tbl
2019-10-02 23:55:20,687 INFO ftrriage: misc:657 bdsol-aci32-leaf3: RW seg_id:11364 in SUG same as EP
segid:11364
2019-10-02 23:55:26,982 INFO ftrriage: main:961 Packet is Exiting fabric with peer-device: bdsol-aci32-
n3k-3 and peer-port: Ethernet1/16

```

This flow is allowed (unexpected) by zoning-rule 4341. The zoning-rules must now be analyzed to understand why.

### Datapath validation — zoning-rules

The zoning-rules corresponding to the last 2 tests are below:

- Expected — flow hits zoning-rule line 4336 (ICMP2 contract).
- Unexpected — flow hits zoning-rule line 4341 (ICMP1 contract).

```

bdsol-aci32-leaf3# show zoning-rule scope 2097154
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir | operSt | Scope | Name | Action | Priority |
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 4326 | 0 | 0 | implicit | uni-dir | enabled | 2097154 | | deny,log | any_any_any(21) |
| 4335 | 0 | 16387 | implicit | uni-dir | enabled | 2097154 | | permit | any_dest_any(16) |
| 4334 | 0 | 0 | implarp | uni-dir | enabled | 2097154 | | permit | any_any_filter(17) |
| 4333 | 0 | 15 | implicit | uni-dir | enabled | 2097154 | | deny,log | any_vrf_any_deny(22) |
| 4332 | 0 | 16386 | implicit | uni-dir | enabled | 2097154 | | permit | any_dest_any(16) |
| 4339 | 32770 | 15 | 5 | uni-dir | enabled | 2097154 | ICMP2 | permit | fully_qual(7) |
| 4341 | 49153 | 32770 | 5 | uni-dir | enabled | 2097154 | ICMP2 | permit | fully_qual(7) |
| 4337 | 32771 | 15 | 5 | uni-dir | enabled | 2097154 | ICMP1 | permit | fully_qual(7) |
| 4336 | 49153 | 32771 | 5 | uni-dir | enabled | 2097154 | ICMP1 | permit | fully_qual(7) |
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Both flows derive the src pcTag of 49153. This is the pcTag of the VRF. This can be verified in the UI:

## Verification the pcTag of the VRF

The screenshot shows the Cisco APIC interface for the 'Prod' tenant. The 'Operational' tab is selected, and the 'VRFs' sub-tab is active. A table lists the VRF configuration:

VRF Name	VRF Alias	Class ID	Segment ID	Scope
VRF1		49153	2097154	2097154

The following happens when the 0.0.0.0/0 prefix is in use with an L3Out:

- Traffic from an internal EPG to an L3Out EPG with 0.0.0.0/0 will derive a destination pcTag of 15.
- Traffic from an L3Out EPG with 0.0.0.0/0 to an ACI internal EPG will derive a source pcTag of the VRF (49153).

The `contract_parser` script gives a holistic view of the zoning-rules:

```

bdsol-aci32-leaf3# contract_parser.py --vrf Prod:VRF1
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-14] dst-epg [dst-14] [flags][contract:{str}] [hit=count]
[7:4339] [vrf:Prod:VRF1] permit ip icmp tn-Prod/ap-App/epg-EPG1(32770) pfx-0.0.0.0/0(15) [contract:uni/tn-Prod/brc-ICMP2] [hit=0]
[7:4337] [vrf:Prod:VRF1] permit ip icmp tn-Prod/ap-App/epg-EPG2(32771) pfx-0.0.0.0/0(15) [contract:uni/tn-Prod/brc-ICMP] [hit=0]
[7:4341] [vrf:Prod:VRF1] permit ip icmp tn-Prod/vrf-VRF1(49153) tn-Prod/ap-App/epg-EPG1(32770) [contract:uni/tn-Prod/brc-ICMP2] [hit=270]
[7:4336] [vrf:Prod:VRF1] permit ip icmp tn-Prod/vrf-VRF1(49153) tn-Prod/ap-App/epg-EPG2(32771) [contract:uni/tn-Prod/brc-ICMP] [hit=0]

```

### Confirming pcTag used by the packet using ELAM Assistant app

The ELAM Assistant App gives another method to confirm the source and destination pcTag of live traffic flows.

The screen shot below shows the ELAM result for traffic from pcTag 32771 to pcTag 49153.

#### ELAM Assistant app output for src 32771 to dst 49153

Packet Forwarding Information	
<b>Forward Result</b>	
Destination Type	To a local port
Destination Logical Port	Po1
Destination Physical Port	eth1/12
Sent to SUP/CPU instead	no
SUP Redirect Reason (SUP code)	NONE
<b>Contract</b>	
Destination EPG pcTag (dclass)	32771 (Prod:App:EPG2)
Source EPG pcTag (sclass)	49153 (Prod:VRF1:l3out-BGP:vlan-2551)

### Conclusion

The usage of 0.0.0.0/0 must be carefully tracked within a VRF as every L3Out using that subnet will inherit the contracts applied to every other L3Out using it. This will likely lead to unplanned permit flows.

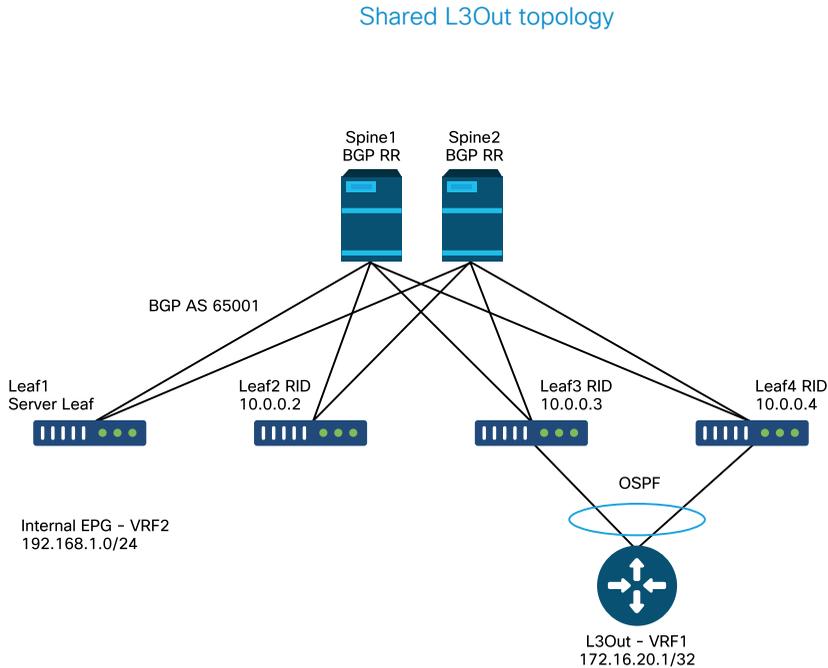
# Shared L3Out

## Overview

This section will discuss how to troubleshoot route-advertisement in Shared L3Out configurations. The term 'Shared L3Out' refers to the scenario where an L3Out is in one VRF but an internal EPG having a contract with the L3Out is in a another VRF. With Shared L3Outs, the route-leaking is being done internally to the ACI fabric.

This section will not go into deep detail about security policy troubleshooting. For that refer to the "Security Policies" chapter of this book. This section will also not talk in detail about External Policy Prefix classification for security purposes. Refer to the "Contract and L3Out" section in the "external forwarding" chapter.

This section uses the following topology for our examples.



At a high level, the following configurations must be in place for a Shared L3Out to function:

- An L3Out subnet must be configured with the 'Shared Route Control Subnet' scope to leak external routes into internal VRFs. 'Aggregate Shared' option can also be selected to leak all routes that are more specific than the configured subnet.
- An L3Out subnet must be configured with the 'Shared Security Import Subnet' scope to program the security policies necessary to allow communication through this L3Out.

- The internal BD subnet must be set to 'Shared between VRFs' and 'Advertise Externally' to program the BD subnet in the external VRF and advertise it.
- A 'tenant' or 'global' scope contract must be configured between the internal EPG and the external EPG of the shared L3Out.

The next section will go into detail about how leaked routes are advertised and learned in ACI.

## Shared L3Out workflow — learning external routes

This section will outline the path of a learned external route as it is advertised into the fabric.

### External route as seen on the border leaf

This command will show the external route learned from OSPF:

```
leaf103# show ip route 172.16.20.1/32 vrf Prod:Vrf1
IP Route Table for VRF "Prod:Vrf1"
'*' denotes best ucast next-hop
 '**' denotes best mcast next-hop
 '[x/y]' denotes [preference/metric]
 '%<string>' in via output denotes VRF <string>

172.16.20.1/32, ubest/mbest: 1/0
   *via 10.10.34.3, vlan347, [110/20], 03:59:59, ospf-default, type-2
```

Next, the route must be imported into BGP. By default, all external routes should be imported into BGP.

### BGP verifications on the border leaf

The route must be in the BGP VPNv4 Address-family with a route-target to be distributed throughout the fabric. The route-target is a BGP extended community exported by the external VRF and imported by any internal VRFs that needs to receive the path.

Next, verify the route-target that is being exported by the external VRF on the BL.

```
leaf103# show bgp process vrf Prod:Vrf1

Information regarding configured VRFs:

BGP Information for VRF Prod:Vrf1
VRF Type                : System
VRF Id                  : 85
VRF state                : UP
VRF configured          : yes
VRF refcount            : 1
VRF VNID                : 2392068
Router-ID               : 10.0.0.3
Configured Router-ID   : 10.0.0.3
Confed-ID               : 0
Cluster-ID             : 0.0.0.0
MSITE Cluster-ID       : 0.0.0.0
No. of configured peers : 1
No. of pending config peers : 0
No. of established peers : 0
VRF RD                  : 101:2392068
VRF EVPN RD            : 101:2392068

...

Wait for IGP convergence is not configured
Export RT list:
65001:2392068
Import RT list:
65001:2392068
Label mode: per-prefix
```

The above output shows that any paths advertised from the external VRF into VPNv4 should receive a route-target of 65001:2392068.

Next, verify the bgp path:

```
leaf103# show bgp ipv4 unicast 172.16.20.1/32 vrf Prod:Vrf1
BGP routing table information for VRF Prod:Vrf1, address family IPv4 Unicast
BGP routing table entry for 172.16.20.1/32, version 30 dest ptr 0xa6f25ad0
Paths: (2 available, best #1)
Flags: (0x80c0002 00000000) on xmit-list, is not in urib, exported
      vpn: version 17206, (0x100002) on xmit-list
Multipath: eBGP iBGP
```

```

Advertised path-id 1, VPN AF advertised path-id 1
Path type: redist 0x408 0x1 ref 0 adv path ref 2, path is valid, is best path
AS-Path: NONE, path locally originated
 0.0.0.0 (metric 0) from 0.0.0.0 (10.0.0.3)
  Origin incomplete, MED 20, localpref 100, weight 32768
  Extcommunity:
    RT:65001:2392068
    VNID:2392068
    COST:pre-bestpath:162:110

VRF advertise information:
Path-id 1 not advertised to any peer

VPN AF advertise information:
Path-id 1 advertised to peers:
 10.0.64.64      10.0.72.66
Path-id 2 not advertised to any peer

```

The above output shows that the path has the correct route-target. The VPNv4 path can also be verified by using 'show bgp vpnv4 unicast 172.16.20.1 vrf overlay-1' command.

## Verifications on the server leaf

For the internal EPG leaf to install the BL-advertised route, it must import the route-target (mentioned above) into the internal VRF. The internal VRF's BGP process can be checked to validate this:

```

leaf101# show bgp process vrf Prod:Vrf2

Information regarding configured VRFs:

BGP Information for VRF Prod:Vrf2
VRF Type           : System
VRF Id             : 54
VRF state          : UP
VRF configured     : yes
VRF refcount       : 0
VRF VNID           : 2916352
Router-ID          : 192.168.1.1
Configured Router-ID : 0.0.0.0
Confed-ID          : 0
Cluster-ID         : 0.0.0.0
MSITE Cluster-ID   : 0.0.0.0
No. of configured peers : 0
No. of pending config peers : 0
No. of established peers : 0
VRF RD             : 102:2916352
VRF EVPN RD        : 102:2916352

```

```

...
Wait for IGP convergence is not configured
Import route-map 2916352-shared-svc-leak
Export RT list:
  65001:2916352
Import RT list:
  65001:2392068
  65001:2916352

```

The above output shows the internal VRF importing the route-target that is exported by the external VRF. Additionally, there is an 'Import Route-Map' that is referenced. The import route-map includes the specific prefixes that are defined in the shared L3Out with the 'Shared Route Control Subnet' flag.

The route-map contents can be checked to ensure it includes the external prefix:

```

leaf101# show route-map 2916352-shared-svc-leak
route-map 2916352-shared-svc-leak, deny, sequence 1
Match clauses:
  pervasive: 2
Set clauses:
route-map 2916352-shared-svc-leak, permit, sequence 2
Match clauses:
  extcommunity (extcommunity-list filter): 2916352-shared-svc-leak
Set clauses:
route-map 2916352-shared-svc-leak, permit, sequence 1000
Match clauses:
  ip address prefix-lists: IPv4-2392068-16387-5511-2916352-shared-svc-leak
  ipv6 address prefix-lists: IPv6-deny-all
Set clauses:

```

```

a-leaf101# show ip prefix-list IPv4-2392068-16387-5511-2916352-shared-svc-leak
ip prefix-list IPv4-2392068-16387-5511-2916352-shared-svc-leak: 1 entries
seq 1 permit 172.16.20.1/32

```

The above output shows the import route-map which includes the subnet to be imported.

The final verifications include checking that the route is in the BGP table and that it is installed in the routing table.

BGP table on server leaf:

```
leaf101# show bgp ipv4 unicast 172.16.20.1/32 vrf Prod:Vrf2
BGP routing table information for VRF Prod:Vrf2, address family IPv4 Unicast
BGP routing table entry for 172.16.20.1/32, version 3 dest ptr 0xa763add0
Paths: (2 available, best #1)
Flags: (0x08001a 00000000) on xmit-list, is in urib, is best urib route, is in HW
  vpn: version 10987, (0x100002) on xmit-list
Multipath: eBGP iBGP

  Advertised path-id 1, VPN AF advertised path-id 1
  Path type: internal 0xc000018 0x40 ref 56506 adv path ref 2, path is valid, is best path
    Imported from 10.0.72.64:5:172.16.20.1/32
  AS-Path: NONE, path sourced internal to AS
    10.0.72.64 (metric 3) from 10.0.64.64 (192.168.1.102)
      Origin incomplete, MED 20, localpref 100, weight 0
      Received label 0
      Received path-id 1
      Extcommunity:
        RT:65001:2392068
        VNID:2392068
        COST:pre-bestpath:162:110
      Originator: 10.0.72.64 Cluster list: 192.168.1.102
```

The route is imported into the internal VRF BGP table and has the expected route-target.

The installed routes can be verified:

```
leaf101# vsh -c "show ip route 172.16.20.1/32 detail vrf Prod:Vrf2"
IP Route Table for VRF "Prod:Vrf2"
'*' denotes best ucast next-hop
'***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>
172.16.20.1/32, ubest/mbest: 2/0
  *via 10.0.72.64%overlay-1, [200/20], 01:00:51, bgp-65001, internal, tag 65001 (mpls-vpn)
    MPLS[0]: Label=0 E=0 TTL=0 S=0 (VPN)
    client-specific data: 548
    recursive next hop: 10.0.72.64/32%overlay-1
    extended route information: BGP origin AS 65001 BGP peer AS 65001 rw-vnid: 0x248004 table-id: 0x36 rw-
mac: 0
  *via 10.0.72.67%overlay-1, [200/20], 01:00:51, bgp-65001, internal, tag 65001 (mpls-vpn)
    MPLS[0]: Label=0 E=0 TTL=0 S=0 (VPN)
    client-specific data: 54a
    recursive next hop: 10.0.72.67/32%overlay-1
    extended route information: BGP origin AS 65001 BGP peer AS 65001 rw-vnid: 0x248004 table-id: 0x36 rw-
mac: 0
```

The above output uses a specific 'vsh -c' command to get the 'detail' output. The 'detail' flag includes the rewrite VXLAN VNID. This is the VXLAN VNID of the external VRF. When the BL receives dataplane traffic with this VNID, it knows to make the forwarding decision in the external VRF.

The rw-vnid value is in hex, so converting to decimal will get the VRF VNID of 2392068. Search for the corresponding VRF using 'show system internal epm vrf all | grep 2392068' on the leaf. A global search can be performed on an APIC using the 'moquery -c fvCtx -f 'fv.Ctx.seg=="2392068"' command.

The next-hop's IP should also point to the BL PTEPs and the '%overlay-1' indicates that the route lookup for the next-hop is in the overlay VRF.

## Shared L3Out workflow — advertising internal routes

As in previous sections, advertising internal BD subnets out a shared L3Out is handled by the following:

- The BD subnet (internal VRF) is installed on the BL (external VRF) as a static route. This static route deployment is a result of the contract relationship between the internal EPG and the L3Out.
- The static route is redistributed into the external protocol when the 'Advertised Externally' scope is set on the BD subnet.

### Verify BD static route on the BL

```
leaf103# vsh -c "show ip route 192.168.1.0 detail vrf Prod:Vrf1"
IP Route Table for VRF "Prod:Vrf1"
'*' denotes best ucast next-hop
'***' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

192.168.1.0/24, ubest/mbest: 1/0, attached, direct, pervasive
  *via 10.0.120.34%overlay-1, [1/0], 00:55:27, static, tag 4294967292
    recursive next hop: 10.0.120.34/32%overlay-1
    vrf crossing information: VNID:0x2c8000 ClassId:0 Flush#:0
```

Notice that in the above output the VNID of the internal VRF is set for the rewrite. The next-hop is also set to the proxy-v4-anycast address.

The above route is advertised externally through the same route-maps that are demonstrated in the "Route Advertisement" section.

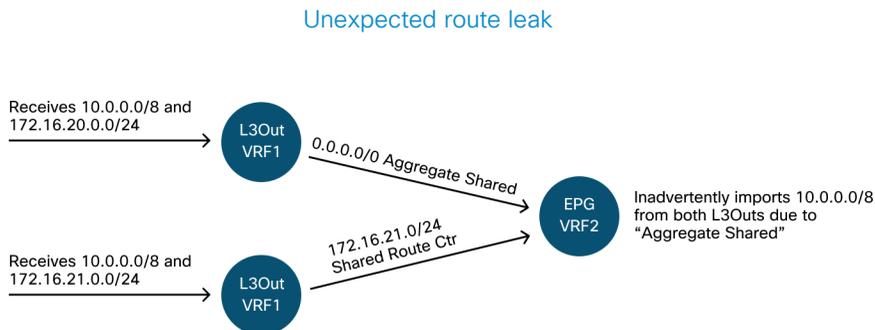
If a BD subnet is set to 'Advertise Externally', it is redistributed into **every L3Out's external protocol** that the internal EPG has a contract relationship with.

## Shared L3Out troubleshooting scenario – unexpected route leaking

This scenario has multiple L3Outs in the external VRF and an internal EPG is receiving a route from an L3Out where the network **is not** defined with the 'shared' scope options.

### Usage of 'Aggregate Shared'

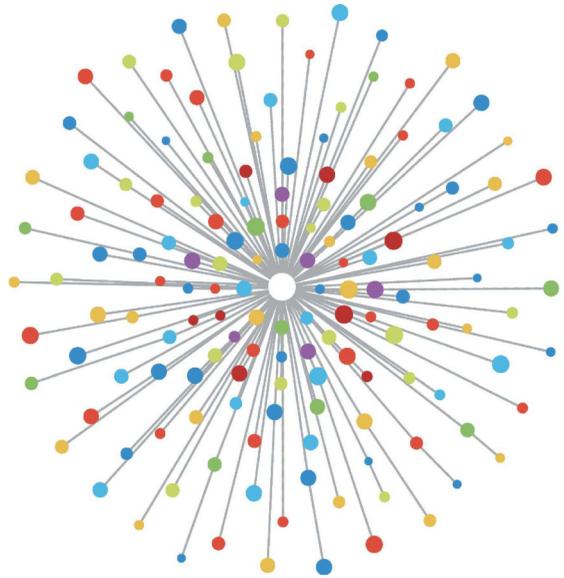
Consider the following figure:



The BGP import-map with the prefix-list programmed from the 'Shared Route Control Subnet' flags is applied at the VRF level. If one L3Out in VRF1 has a subnet with 'Shared Route Control Subnet', then all routes received on L3Outs within VRF1 that match this Shared Route Control Subnet will get imported into VRF2.

The above design can result in unexpected traffic flows. If there are no contracts between the internal EPG and the unexpected advertising L3Out EPG, then there will be traffic drops.

# VMM integration



## Overview

ACI controllers have the capability to integrate with third-party virtual machine managers (VMMs).

This is one of the key features of ACI as it simplifies and automates operations for end-to-end networking configuration of the fabric and to workloads that connect to it. ACI offers a single overlay policy model that can be extended across multiple workload types, i.e. virtual machines, bare metal servers, and containers.

This chapter will specifically focus on some typical troubleshooting scenarios related to the VMware vCenter VMM integration.

The reader will walk through:

- Investigation on vCenter communication faults.
- Host and VM dynamic discovery process and failure scenarios.
- Hypervisor Load Balancing algorithms.

# vCenter connectivity

## Role-Based Access Control (RBAC)

The mechanisms by which APIC is able to interface with the vCenter Controller are dependent on the user account associated to a given VMM Domain. Specific requirements are outlined for the vCenter user associated with the VMM Domain to ensure that the APIC can successfully perform operations on the vCenter, whether it is pushing and retrieving inventory and configurations or monitoring and listening to managed inventory related events.

The easiest way to remove concern about such requirements is to use the administrator vCenter account that has full access; however, this kind of freedom is not always available to the ACI administrator.

The minimum privileges for a custom user account, as of ACI version 4.2, are as follows:

- **Alarms**
  - APIC creates two alarms on the folder. One for DVS and another for port group. An alarm is raised when the EPG or VMM Domain policy is deleted on the APIC, however vCenter is unable to delete the corresponding Port Group or DVS due to having VMs attached to it.
- **Distributed Switch**
- **dvPort Group**
- **Folder**
- **Network**
  - APIC manages the network settings such as add or delete port groups, setting host/DVS MTU, LLDP/CDP, LACP etc.

- **Host**
  - If using AVS in addition to above, the user needs the Host privilege on the datacenter where APIC will create DVS.
  - **Host.Configuration.Advanced settings**
  - **Host.Local operations.Reconfigure virtual machine**
  - **Host.Configuration.Network configuration**
    - This is needed for AVS and the auto-placement feature for virtual Layer 4 to Layer 7 Service VMs. For AVS, APIC creates VMK interface and places it in VTEP port group which is used for OpFlex.
- **Virtual machine**
  - If Service Graphs are in use, the Virtual machine privilege for the virtual appliances is also required.
  - **Virtual machine.Configuration.Modify device settings**
  - **Virtual machine.Configuration.Settings**

## Troubleshooting RBAC-related issues

RBAC issues are most often encountered during initial setup of a VMM Domain but could be encountered if a vCenter administrator were to modify permissions of the user account associated with the VMM Domain after initial setup has already taken place.

The symptom can present itself in the following ways:

- Partial or complete inability to deploy new services (DVS creation, port group creation, some objects are successfully deployed but not all).

- Operational inventory is incomplete or missing from ACI administrator views.
- Faults raised for unsupported vCenter operation, or for any of the scenarios above (e.g. port group deployment failure).
- vCenter controller is reported as offline and faults indicate that there is connectivity or credential related issues.

### Solution for RBAC-related issues

Verify all the above permissions are granted to the vCenter user that is configured in the VMM Domain.

Another method is to login directly to the vCenter with the same credentials as defined in the VMM Domain configuration and attempt similar operations (port group creation, etc.). If the user is not able to perform these same operations while logged in directly to the vCenter, clearly the correct permissions are not granted to the user.

## Connectivity

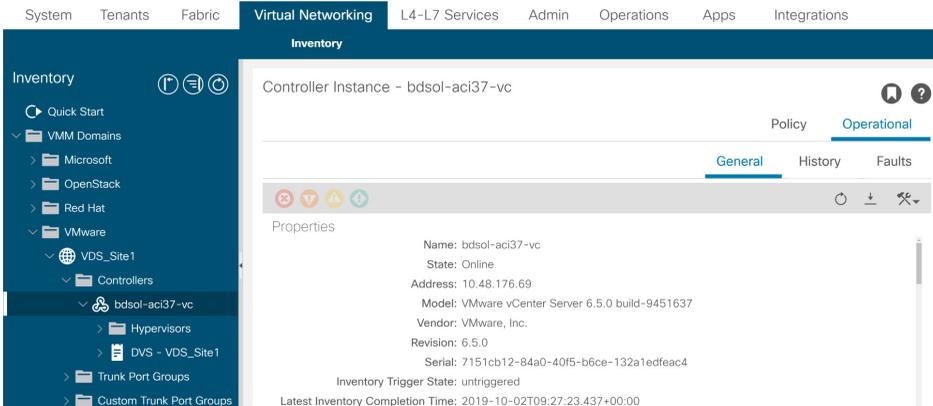
When troubleshooting a VMM connectivity related issue, it is important to note some of the fundamental behaviors of how ACI communicates with vCenter.

The first and most pertinent behavior is that only one APIC in the cluster is sending configuration and collecting inventory at any given point. This APIC is referred to as the **shard leader** for this VMM Domain. However, multiple APICs are listening for **vCenter Events** in order to account for a scenario where the shard leader missed an event for any reason. Following the same distributed architecture of APICs, a given VMM Domain will have one APIC handling primary data and functionality (in this case, the shard leader), and two replicas (in the case of VMM they are referred to as **followers**). To distribute the handling of VMM communication and functionality across APICs, any two VMM Domains can either have the same or a different shard leaders.

### Connectivity troubleshooting

The vCenter connectivity state can be found by navigating to the VMM controller of interest in the GUI or using the CLI command listed below.

#### VMWare VMM Domain - vCenter connectivity state



```

apic2# show vmware domain name VDS_Site1 vcenter 10.48.176.69
Name                : bdsol-aci37-vc
Type                : vCenter
Hostname or IP     : 10.48.176.69
Datacenter         : Site1
DVS Version        : 6.0
Status           : online
Last Inventory Sync : 2019-10-02 09:27:23
Last Event Seen    : 1970-01-01 00:00:00
Username           : administrator@vsphere.local
Number of ESX Servers : 2
Number of VMs      : 2
Faults by Severity  : 0, 0, 0, 0
Leader             : bdsol-aci37-apic1

Managed Hosts:
ESX      VMs  Adjacency  Interfaces
-----
10.48.176.66  1   Direct    leaf-101 eth1/11, leaf-102 eth1/11
10.48.176.67  1   Direct    leaf-301 eth1/11, leaf-302 eth1/11
    
```

If a VMM controller is indicated to be offline, a fault will be thrown similar to below:

```
Fault fltCompCtrlrConnectFailed
Rule ID:130
Explanation:
This fault is raised when the VMM Controller is marked offline. Recovery is in process.
Code: F0130
Message: Connection to VMM controller: host0rIp with name name in datacenter rootContName in domain:
domName is failing repeatedly with error: [remoteErrMsg]. Please verify network connectivity of VMM
controller host0rIp and check VMM controller user credentials are valid.
```

**1. The first step in troubleshooting a connectivity issue between the APIC and vCenter is understanding which APIC is the shard leader for the given VMM Domain. The easiest way to determine this information is to run the command 'show vmware domain name <domain>' on any APIC.**

```
apic1# show vmware domain name VDS_Site1
Domain Name                : VDS_Site1
Virtual Switch Mode        : VMware Distributed Switch
Vlan Domain                : VDS_Site1 (1001-1100)
Physical Interfaces        : leaf-102 eth1/11, leaf-301 eth1/11, leaf-302 eth1/11,
                           leaf-101 eth1/11
Number of EPGs             : 2
Faults by Severity         : 0, 0, 0, 0
LLDP override              : RX: enabled, TX: enabled
CDP override               : no
Channel Mode override      : mac-pinning
NetFlow Exporter Policy    : no
Health Monitoring          : no
vCenters:
Faults: Grouped by severity (Critical, Major, Minor, Warning)
vCenter                    Type      Datacenter    Status   ESXs   VMs   Faults
-----
10.48.176.69              vCenter Site1          online   2      2     0,0,0,0
APIC Owner:
Controller  APIC      Ownership
-----
bdsol-    apic1    Leader
aci37-vc
bdsol-    apic2    NonLeader
aci37-vc
bdsol-    apic3    NonLeader
aci37-vc
```

**2. After identifying the APIC which is actively communicating with the vCenter, verify IP connectivity with tools such as ping.**

```

apic1# ping 10.48.176.69
PING 10.48.176.69 (10.48.176.69) 56(84) bytes of data.
 64 bytes from 10.48.176.69: icmp_seq=1 ttl=64 time=0.217 ms
 64 bytes from 10.48.176.69: icmp_seq=2 ttl=64 time=0.274 ms
 64 bytes from 10.48.176.69: icmp_seq=3 ttl=64 time=0.346 ms
 64 bytes from 10.48.176.69: icmp_seq=4 ttl=64 time=0.264 ms
 64 bytes from 10.48.176.69: icmp_seq=5 ttl=64 time=0.350 ms
^C
--- 10.48.176.69 ping statistics ---
 5 packets transmitted, 5 received, 0% packet loss, time 4084ms
 rtt min/avg/max/mdev = 0.217/0.290/0.350/0.052 ms

```

If the vCenter was configured using the FQDN rather than IP address, the nslookup command can be used to verify name resolution.

```

apic1:>> nslookup bdsol-aci37-vc
Server: 10.48.37.150
Address: 10.48.37.150#53
Non-authoritative answer:
Name: bdsol-aci37-vc.cisco.com
Address: 10.48.176.69

```

**3. Check the APIC routing table to verify if out-of-band or in-band is preferred for connectivity and which gateway is used:**

```

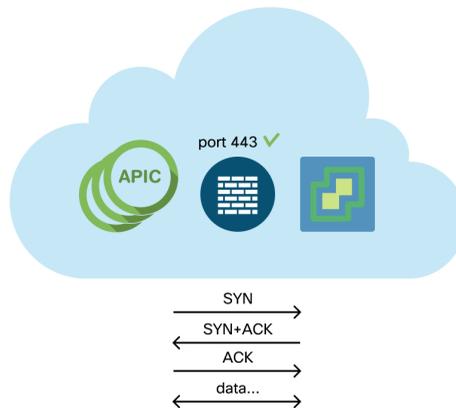
apic1# bash
admin@apic1:>> route
Kernel IP routing table

```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
default	10.48.176.1	0.0.0.0	UG	16	0	0	oobmgmt

#### 4. Ensure Port 443 is allowed between all APICs and the vCenter, including any firewalls in the path of communication.

vCenter <-> APIC - HTTPS (TCP port 443) - communication



General HTTPS reachability from the APICs to vCenter can be tested with a curl:

```
apic2# curl -v -k https://10.48.176.69
* Rebuilt URL to: https://10.48.176.69/
* Trying 10.48.176.69...
* TCP_NODELAY set
* Connected to 10.48.176.69 (10.48.176.69) port 443 (#0)
...
```

Verify that the shard leader has an established TCP connection on port 443 using the netstat command.

```
apic1:>> netstat -tulaen | grep 10.48.176.69
tcp 0 0 10.48.176.57:40806 10.48.176.69:443 ESTABLISHED 600 13062800
```

5. If possible, perform a packet capture along the path in between the shard leader and vCenter in an effort to identify if traffic is being sent and received by either device.

## VMware inventory

The following table shows a list of VMWare VDS parameters and specifies whether these are configurable by the APIC.

VMware VDS parameters managed by APIC

VMware VDS	Default Value	Configurable Using Cisco APIC Policy?
<b>Name</b>	VMM Domain name	Yes (Derived from Domain)
<b>Description</b>	'APIC Virtual Switch'	No
<b>Folder Name</b>	VMM Domain name	Yes (Derived from Domain)
<b>Version</b>	Highest supported by vCenter	Yes
<b>Discovery Protocol</b>	LLDP	Yes
<b>Uplink Ports and Uplink Names</b>	8	Yes (From Cisco APIC Release 4.2(1))
<b>Uplink Name Prefix</b>	uplink	Yes (From Cisco APIC Release 4.2(1))
<b>Maximum MTU</b>	9000	Yes
<b>LACP policy</b>	disabled	Yes
<b>Port mirroring</b>	0 sessions	Yes
<b>Alarms</b>	2 alarms added at the folder level	No

The following table shows a list of VMWare VDS Port Group parameters and specifies whether these are configurable by the APIC.

## VMWare VDS Port Group parameters managed by APIC

VMware VDS Port Group	Default Value	Configurable using APIC Policy
<b>Name</b>	Tenant Name   Application Profile Name   EPG Name	Yes (Derived from EPG)
<b>Port binding</b>	Static binding	No
<b>VLAN</b>	Picked from VLAN pool	Yes
<b>Load balancing algorithm</b>	Derived based on port-channel policy on APIC	Yes
<b>Promiscuous mode</b>	Disabled	Yes
<b>Forged transmit</b>	Disabled	Yes
<b>MAC change</b>	Disabled	Yes
<b>Block all ports</b>	FALSE	No

**Inventory troubleshooting:**

Inventory sync events occur to ensure that the APIC is aware of vCenter events that may require the APIC to dynamically update policy. There are two types of inventory sync events that can occur between vCenter and the APIC; a full inventory sync and an event-based inventory sync. The default schedule of a full inventory sync between the APIC and vCenter is every 24 hours, however these can also be manually triggered. Event-based inventory syncs are typically tied to triggered tasks, such as a vMotion. In this scenario, if a virtual machine moves from one host to another, and those hosts are connected to two different leaf switches, the APIC will listen for the VM migration event and, in the scenario of on-demand deployment immediacy, unprogram the EPG on the source leaf, and program the EPG on the destination leaf.

Depending on the deployment immediacy of EPGs associated with a VMM domain, failure to pull inventory from the vCenter could have undesirable consequences. In the scenario that inventory has failed to complete or is partial,

there will always be a fault raised indicating the object or objects which have caused the failure.

### Scenario 1 - Virtual machine with invalid backing:

If a virtual machine is moved from one vCenter to another, or the virtual machine is determined to have an invalid backing (e.g. a port group attachment to an old/deleted DVS), the vNIC will be reported to have an operational issues.

```
Fault fltCompVnicOperationalIssues
Rule ID:2842
Explanation:
This fault is raised when ACI controller failed to update the properties of a VNIC (e.g., it can not find the EPG that the VNIC attached to).
Code: F2842
Message: Operational issues detected for VNic name on VM name in VMM controller: hostOrIp with name name in datacenter rootContName in domain: domName due to error: issues.
```

### Resolution:

Remediate the virtual machines indicated in the fault by assigning a valid port group on the affected vNIC of the VM.

### Scenario 2 – vCenter administrator modified a VMM managed object on the vCenter:

Modifying objects which are managed by the APIC from vCenter is not a supported operation. The following fault would be seen if an unsupported operation is performed on vCenter.

```
Fault fltCompCtrlrUnsupportedOperation
Rule ID:133
Explanation:
This fault is raised when deployment of given configuration fails for a Controller.
Code: F0133
Message: Unsupported remote operation on controller: hostOrIp with name name in datacenter rootContName in domain domName detected, error: [deployIssues]
```

### Resolution:

If this scenario is encountered, try to undo the unsupported change in vCenter and then trigger an 'inventory sync' manually.

## VMWare VMM Domain - vCenter controller - trigger inventory sync

The screenshot displays the 'Inventory' pane on the left, which is expanded to show the 'VMware' domain. Underneath, the 'VDS\_Site1' domain is expanded to show 'Controllers', and the 'bdsol-aci37-vc' controller instance is selected. A red box highlights the 'Trigger Inventory Sync' button, which is a blue button with white text. To the right of the 'Inventory' pane, the 'Controller Instance - bdsol-aci37-vc' properties are displayed. The properties include: Name: bdsol-aci37-vc, Type: vCenter, Host Name (or IP Address): 10.48.176.69, DVS Version: 6.0.0, Datacenter: Site1, and Stats Collection: Enabled (with a 'Disabled' button next to it).

## VMware DVS version

When creating a new vCenter controller as part of a VMM Domain, the default setting for the DVS Version will be to use the 'vCenter Default'. When selecting this, the DVS version will be created with the version of the vCenter.

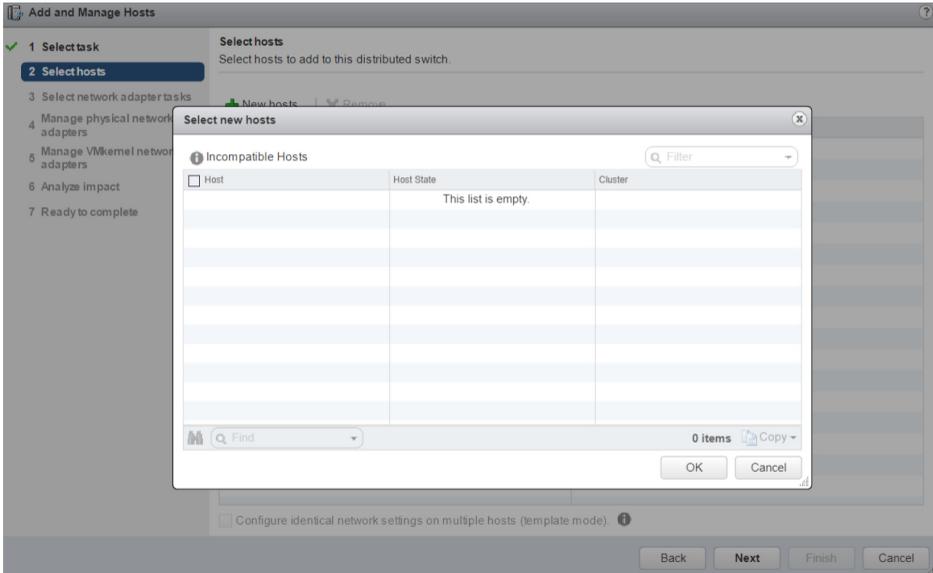
## VMWare VMM Domain - vCenter controller creation

Create vCenter Controller ? ×

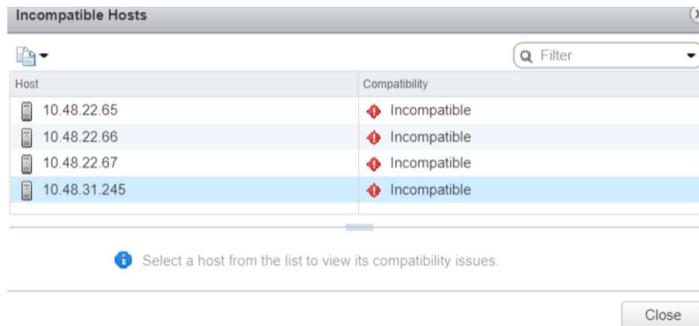
Name:	bdsol-aci20-vc
Host Name (or IP Address):	10.48.33.45
DVS Version:	vCenter Default
Datacenter:	POD20
Stats Collection:	<input checked="" type="radio"/> Enabled <input type="radio"/> Disabled
Management EPG:	select an option
Associated Credential:	bdsol-aci20-vc 

This means that in the example of a vCenter running 6.5 and ESXi servers running 6.0, the APIC will create a DVS with version 6.5 and hence the vCenter administrator will be unable to add the ESXi servers running 6.0 into the ACI DVS.

APIC managed DVS - vCenter host addition - empty list



APIC managed DVS - vCenter host addition - incompatible hosts



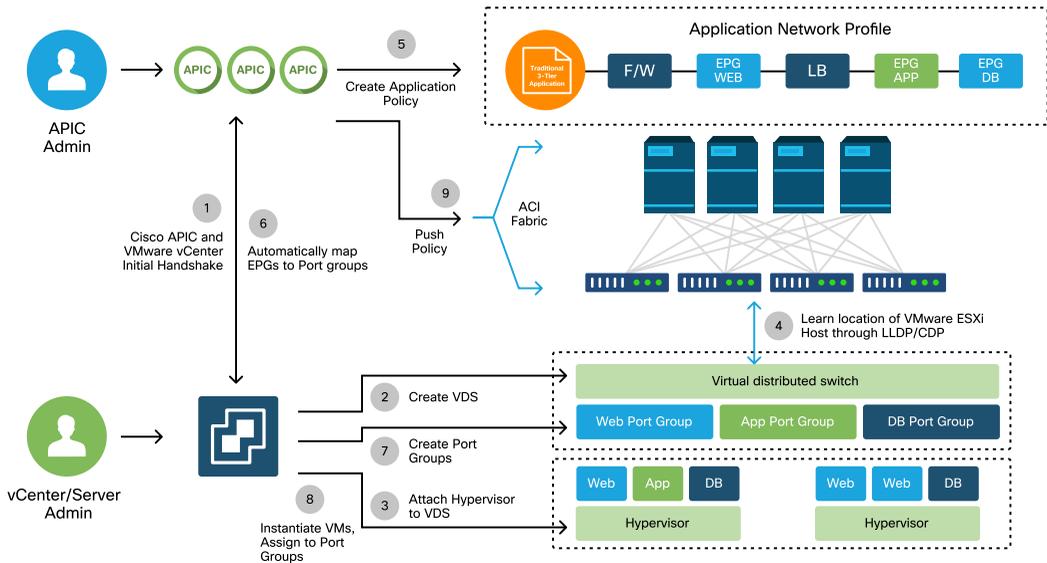
So, when creating a VMM Domain make sure to select the correct 'DVS Version' such that the necessary ESXi servers can be added to the DVS.

# Host dynamic discovery

## Host / VM discovery process

VMM integration in ACI differentiates itself from manual provisioning in that the fabric can dynamically discover where hosts and applicable virtual machines are connected to efficiently deploy policy. Through this dynamic process, ACI can optimize utilization of hardware resources on the leaf switches as VLANs, SVIs, zoning rules, etc. are deployed on nodes only when there is an endpoint connected which requires the policy. The benefit to the network administrator, from an ease of use perspective, is that ACI will provision VLAN/policy where VMs connect in an automated way. To determine where policy must be deployed, the APIC will use information from multiple sources. The following diagram outlines the basic steps of the host discovery process when using a DVS-based VMM Domain.

### VMWare VMM Domain – Deployment workflow



In short the following key steps are happening when:

- LLDP or CDP is exchanged between hypervisor and leaf switches.
- Hosts report adjacency info to vCenter.
- vCenter notifies APIC of adjacency info:
  - APIC knows about host via inventory sync.
- APIC pushes policy to the leaf port:
  - please review "Resolution Immediacy" sub-section within this section to further understand these conditions.
- If vCenter adjacency info is lost, APIC can remove policy.

As can be seen, CDP/LLDP plays a key role in the discovery process and it is important to make sure this is properly configured and both sides are using the same protocol.

## **Fabric LooseNode / intermediate switch - use case**

In a deployment using a blade chassis with an intermediate switch between the leaf switches and the hypervisor, the APIC needs to 'stitch' the adjacency together. In this scenario, multiple discovery protocols could be used as the intermediate switch may have different protocol requirements than the host.

In a setup with a blade server and an intermediate switch (i.e. blade chassis switch), ACI should detect the intermediate switch and map the hypervisors behind it. The intermediate switch is referred to in ACI as a LooseNode or an 'Unmanaged Fabric Node'. The detected LooseNodes can be viewed under 'Fabric > Inventory > Fabric Membership > Unmanaged Fabric Nodes'. By navigating to one of these types of servers in the GUI, the user can view the path from leaf to intermediate switch to host.

## APIC UI – Unmanaged fabric nodes (LooseNodes)

The screenshot displays the APIC (Application Policy Infrastructure Controller) user interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Fabric' tab is selected, and the 'Inventory' sub-tab is active. In the left-hand navigation pane, 'Fabric Membership' is highlighted. The main content area shows 'Fabric Membership' with tabs for 'Registered Nodes', 'Nodes Pending Registration', 'Unreachable Nodes', and 'Unmanaged Fabric Nodes'. The 'Unmanaged Fabric Nodes' tab is selected, displaying a list of nodes. The first node is 'Unmanaged Fabric Node - 10.48.22.80 (bdsol-aci12-ucs-A)'. Below this, the 'General' tab is selected, showing the node's properties and a table of fabric node links.

Properties

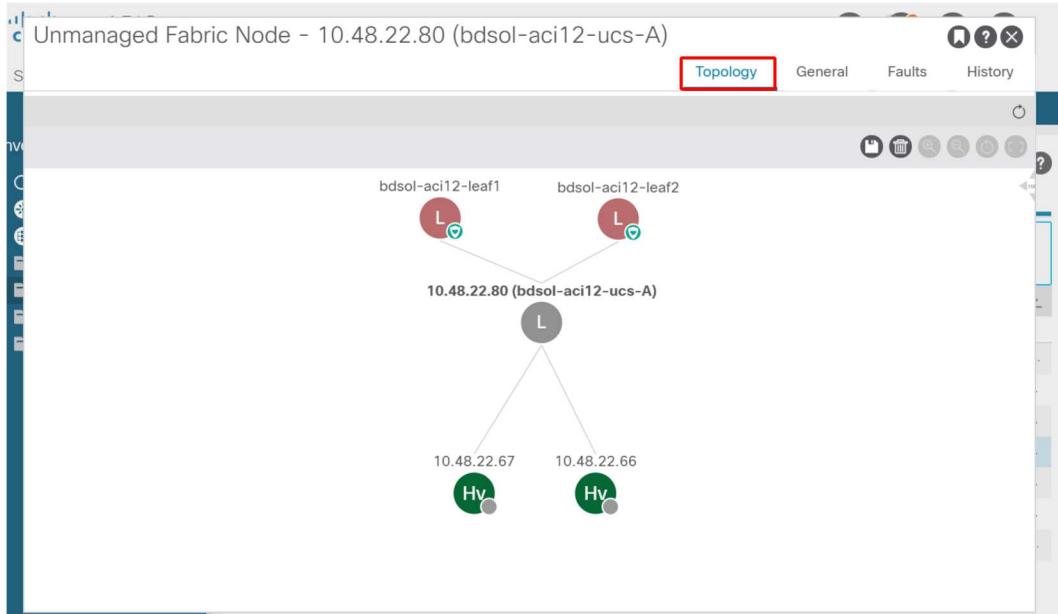
System Name: bdsol-aci12-ucs-A  
 Management IP: 10.48.22.80  
 System Description: Cisco Nexus Operating System (NX-OS) Software 5.0(3)N2(3.11e) TAC support: http://www.cisco.com/tac Copyright (c) 2002-2016, Cisco Systems, Inc. All rights reserved.

Fabric Node Link	Node ID	Node Name	Interface
	101	bdsol-aci12-leaf1	po1
	102	bdsol-aci12-leaf2	po1

With LLDP or CDP discovery in place, ACI can determine the topology for such LooseNodes, given that the hypervisor downstream of the intermediate switch is managed through VMM integration, and the leaf itself has an adjacency to the intermediate switch from downstream.

This concept is illustrated by the image below.

### APIC UI – Unmanaged Fabric Node path

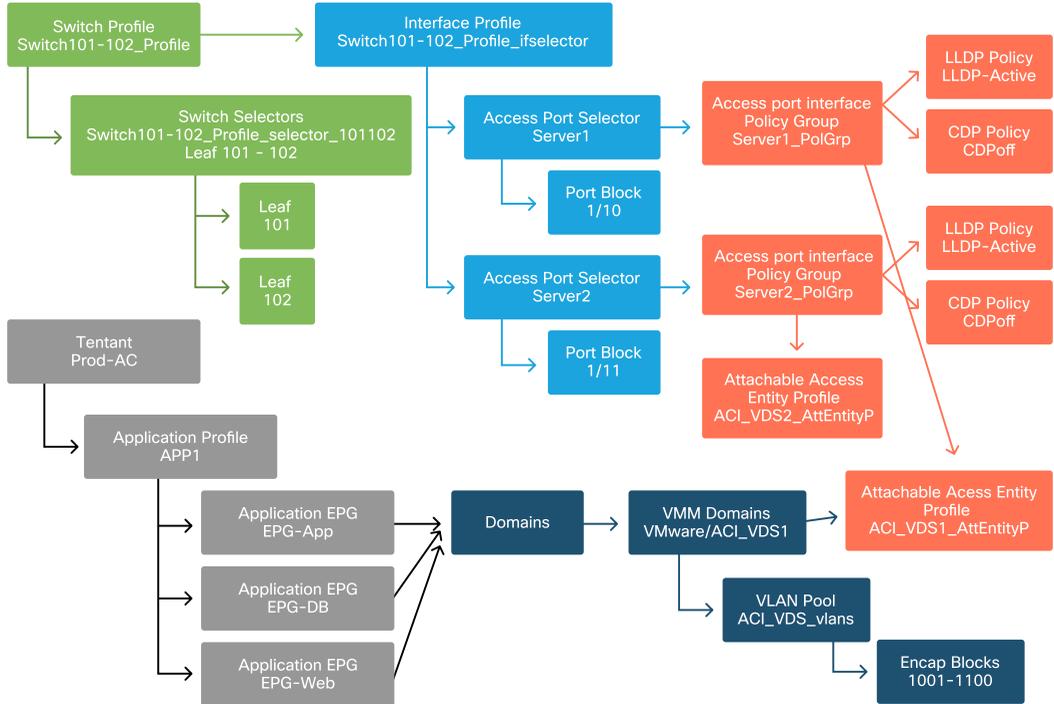


## Resolution Immediacy

In scenarios where critical services utilize the VMM-integrated DVS such as management connectivity to vCenter/ESXi, it is prudent to use the Pre-provision Resolution Immediacy. With this setting, the mechanism of dynamic host discovery is removed and instead policy / VLANs are statically programmed on the host facing interfaces. In this configuration, the VMM VLANs will always be deployed to all interfaces tied to the AEP referenced by the VMM Domain. This removes the possibility that a critical VLAN (such as management) is removed from a port due to a discovery protocol-related adjacency event.

Refer to the below diagram:

Pre-provision deployment example



If Pre-provision was set for an EPG in the ACI\_VDS1 VMM Domain, then VLANs would be deployed on links for Server1 but not Server2 as Server2's AEP does not include the ACI\_VDS1 VMM Domain.

To summarize the resolution immediacy settings:

- On-Demand - Policy is deployed when adjacency is established between leaf and host and a VM attached to the port group.

- Immediate - Policy is deployed when adjacency is established between leaf and host.
- Pre-provision - Policy is deployed to all ports using an AEP with the VMM Domain contained, no adjacency is required.

## Troubleshooting scenarios

### VM cannot resolve ARP for its default gateway

In this scenario, VMM integration has been configured and the DVS has been added to the hypervisor but the VM cannot resolve ARP for its gateway in ACI. For the VM to have network connectivity, verify that adjacency has established and VLANs are deployed.

First, the user can check the leaf has detected the host by using 'show lldp neighbors' or 'show cdp neighbors' on the leaf depending on the protocol selected.

```
Leaf101# show lldp neighbors
Capability codes:
  (R) Router, (B) Bridge, (T) Telephone, (C) DOCSIS Cable Device
  (W) WLAN Access Point, (P) Repeater, (S) Station, (O) Other
Device ID           Local Intf      Hold-time  Capability  Port ID
-----
bdsol-aci37-asic1  Eth1/1         120        B           eth2-1
bdsol-aci37-asic2  Eth1/2         120        B           eth2-1
bdsol-aci37-os1    Eth1/11        180        B           0050.565a.55a7
S1P1-Spine201     Eth1/49        120        BR          Eth1/1
S1P1-Spine202     Eth1/50        120        BR          Eth1/1
Total entries displayed: 5
```

If needed from a troubleshooting perspective, this can be validated from the ESXi side both on the CLI and GUI:

```
[root@host:~] esxcli network vswitch dvs vmware list
VDS_Site1
  Name: VDS_Site1
  ...
  Uplinks: vnic7, vnic6
  VMware Branded: true
```

```
DVPort:
  Client: vmnic6
  DVPortgroup ID: dvportgroup-122
  In Use: true
  Port ID: 0

  Client: vmnic7
  DVPortgroup ID: dvportgroup-122
  In Use: true
  Port ID: 1

[root@host:~] esxcfg-nics -l
Name PCI Driver Link Speed Duplex MAC Address MTU Description
vmnic6 0000:09:00.0 enic Up 10000Mbps Full 4c:77:6d:49:cf:30 9000 Cisco Systems Inc Cisco VIC
Ethernet NIC
vmnic7 0000:0a:00.0 enic Up 10000Mbps Full 4c:77:6d:49:cf:31 9000 Cisco Systems Inc Cisco VIC
Ethernet NIC

[root@host:~] vim-cmd hostsvc/net/query_networkhint --pnic-name=vmnic6 | grep -A2 "System Name"
  key = "System Name",
  value = "Leaf101"
}
```

## vCenter Web Client - host - vmnic LLDP/CDP adjacency details

vmnic6			
All	Properties	CDP	LLDP
<b>Link Layer Discovery Protocol</b>			
Chassis ID			00:3a:9c:45:12:6b
Port ID			Eth1/11
Time to live			109
TimeOut			60
Samples			437068
Management Address			10.48.176.70
Port Description			topology/pod-1/paths-101/pathep-[eth1/11]
System Description			topology/pod-1/node-101
System Name			S1P1-Leaf101
<b>Peer device capability</b>			
Router			Enabled
Transparent bridge			Enabled
Source route bridge			Disabled
Network switch			Disabled
Host			Disabled
IGMP			Disabled
Repeater			Disabled

If the leaf LLDP adjacency cannot be seen from the ESXi host, this is often caused by using a network adapter which is configured to generate LLDPDU's instead of the ESXi OS. Make sure to validate if the network adapter has LLDP enabled and hence is consuming all LLDP information. If this is the case, be sure to disable LLDP on the adapter itself so it is controlled through the vSwitch policy.

Another cause might be that there is a mis-alignment between the discovery protocols used between leaf and ESXi Hypervisor. Make sure on both ends to use the same discovery protocol.

To check if the CDP/LLDP settings are aligned between ACI and the DVS in the APIC UI, navigate to 'Virtual Networking > VMM Domains > VMWare > Policy > vSwitch Policy'. Make sure to only enable either LLDP or CDP policy as they are mutually exclusive.

### APIC UI - VMWare VMM Domain - vSwitch policy

#### Properties

Port Channel Policy:	VDS_lacpLagPol	▼	
LLDP Policy:	LLDP_enabled	▼	
CDP Policy:	CDP_disabled	▼	
NetFlow Exporter Policy:	select an option	▼	

In vCenter go to: 'Networking > VDS > Configure'.

### vCenter Web Client UI - VDS properties

- ⏪
- ▼ Settings
- Properties
- Topology
- Private VLAN
- NetFlow
- Port mirroring
- Health check
- ▼ More
- Network Protocol Profiles
- Resource Allocation

#### Properties

**General**

Name: VDS\_Site1

Manufacturer: VMware, Inc.

Version: 6.0.0

Number of uplinks: 8

Number of ports: 24

Network I/O Control: Disabled

**Description:**

APIC Virtual Switch

**Advanced**

MTU: 9000 Bytes

Multicast filtering mode: Basic

**Discovery protocol**

Type: Link Layer Discovery Protocol

Operation: Both

**Administrator contact**

Name:

Other details:

Correct the LLDP/CDP settings if needed.

Then validate the APIC observes the ESXi host's LLDP/CDP neighborhood against the leaf switch in the UI under 'Virtual Networking' > VMM Domains > VMWare > Policy > Controller > Hypervisor > General'.

### APIC UI - VMWare VMM Domain - Hypervisor details

The screenshot shows the APIC interface with the following details:

- Navigation: System > Tenants > Fabric > **Virtual Networking** > L4-L7 Services > Admin > Operations > Apps > Integrations
- Left Panel: Inventory > VMM Domains > VMWare > VDS\_Site1 > Controllers > bdsol-aci37-vc > Hypervisors > 10.48.176.66
- Main Panel: Hypervisor - 10.48.176.66 (Powered On)
- General Tab:
 

Management Address	Interface Name	Proto	Neighbor ID
10.48.176.70	Pod-1/Node-101/eth1/11	LLDP	00:3a:9c:45:12:6b
10.48.176.71	Pod-1/Node-102/eth1/11	LLDP	00:3a:9c:1b:37:4b

Name	MAC	State	IP Address
vmk0	28.AC.9E.DE.D2.7A	Up	10.48.176.66

If this is showing expected values, then the user can validate that the VLAN is present on the port toward the host.

```
S1P1-Leaf101# show vlan encap-id 1035
```

```

VLAN Name                Status  Ports
-----
12  Ecommerce:Electronics:APP  active  Eth1/11

VLAN Type  Vlan-mode
-----
12  enet  CE

```

### **vCenter/ESXi management VMK attached to APIC-pushed DVS**

In a scenario where vCenter or ESXi management traffic needs to utilize the VMM integrated DVS, it is important to take some extra care to avoid a stalemate in activating the dynamic adjacencies and activate the required VLANs.

For vCenter, which is typically built before VMM integration is configured, it is important to use a physical domain and static path to assure the vCenter VM's encapsulation VLAN is always programmed on the leaf switches so that it can be used before VMM integration is fully set up. Even after setting up the VMM integration, it is advised to leave this static path in place to always assure availability of this EPG.

For the ESXi hypervisors, as per the "Cisco ACI Virtualization Guide" on Cisco.com, when migrating onto the vDS it is important to make sure that the EPG where the VMK interface will be connected is deployed with the resolution immediacy set to Pre-provision. This will make sure the VLAN is always programmed on the leaf switches without relying on LLDP/CDP discovery of the ESXi hosts.

### **Host adjacencies not discovered behind LooseNode**

Typical causes of LooseNode discovery issues are:

- CDP/LLDP is not enabled
  - CDP/LLDP must be exchanged between the intermediate switch, the leaf switches and ESXi hosts
  - For Cisco UCS, this is accomplished via a network control policy on the vNIC
- A change in the management IP of the LLDP/CDP neighbor breaks connectivity
  - The vCenter will see the new management IP in the LLDP/CDP adjacency, but will not update APIC
  - Trigger a manual inventory sync to fix

- VMM VLANs are not added to the intermediate switch
  - The APIC doesn't program third party blade/intermediate switches.
  - Cisco UCSM integration app (ExternalSwitch) available in 4.1(1) release.
  - VLANs must be configured and trunked to uplinks connected to ACI leaf nodes and downlinks connected to hosts

### F606391 - Missing adjacencies for the physical adapter on the host

When seeing the fault below:

```
Affected Object: comp/prov-VMware/ctrlr-[DVS-DC1-ACI-LAB]-DVS1/hv-host-104
Fault delegate: [FSM:FAILED]: Get LLDP/CDP adjacency information for the physical adapters on the host: bdsol-aci20-os3 (TASK:ifc:vmmgr:CompHvGetHpNicAdj)
```

Please review the workflow in section "VM cannot resolve ARP for its default gateway" as this means there are missing CDP/LLDP adjacencies. These adjacencies should be verified end-to-end.

## Hypervisor uplink load balancing

When connecting hypervisors such as ESXi to an ACI fabric, they will typically be connected with multiple uplinks. In fact, it is recommended to have an ESXi host connected to at least two leaf switches. This will minimize the impact of failure scenarios or upgrades.

In order to optimize how uplinks are used by the workloads running on a hypervisor, VMware vCenter configurations allow configuring multiple load balancing algorithms for VM-generated traffic towards the hypervisor's uplinks.

It is crucial to have all hypervisors and the ACI fabric aligned with the same load balancing algorithm configuration to ensure correct connectivity is in place. Failure to do so may result in intermittent traffic flow drops and endpoint moves in the ACI fabric.

This can be seen in an ACI fabric by excessive alerts such as:

```
F3083 fault
ACI has detected multiple MACs using the same IP address 172.16.202.237.
MACs: Context: 2981888. fvCEps:
uni/tn-BSE_PROD/ap-202_Voice/epg-VLAN202_Voice/cep-00:50:56:9D:55:B2;
uni/tn-BSE_PROD/ap-202_Voice/epg-VLAN202_Voice/cep-00:50:56:9D:B7:01;
```

or

```
[F1197][raised][bd-limits-exceeded][major][sys/ctx-[vxlan-2818048]/bd-[vxlan-16252885]/fault-F1197]
Learning is disabled on BD Ecommerce:BD01
```

This chapter will cover VMWare ESXi host connectivity into ACI but is applicable for most hypervisors.

## Rack server

When looking at the various ways an ESXi host can connect to an ACI fabric, they are divided in 2 groups, switch dependent and switch independent load balancing algorithms.

Switch independent load balancing algorithms are ways to connect where no specific switch configuration is needed. For switch dependent load balancing, switch-specific configurations are required.

Make sure to validate if vSwitch Policy is in line with the 'ACI Access Policy Group' requirements as per the table below.

## Teaming and ACI vSwitch policy

VMware Teaming and Failover Mode	ACI vSwitch Policy	Description	ACI Access Policy Group - Port Channel Required
<b>Route based on the originating virtual port</b>	MAC Pinning	Select an uplink based on the virtual port IDs on the switch. After the virtual switch selects an uplink for a virtual machine or a VMKernel adapter, it always forwards traffic through the same uplink for this virtual machine or VMKernel adapter.	No
<b>Route based on Source MAC hash</b>	NA	Select an uplink based on a hash of the source MAC address	NA
<b>Explicit Failover Order</b>	Use Explicit Failover Mode	From the list of active adapters, always use the highest order uplink that passes failover detection criteria. No actual load balancing is performed with this option.	No
<b>Link Aggregation(LAG) - IP Hash Based</b>	Static Channel - Mode On	Select an uplink based on a hash of the source and destination IP addresses of each packet. For non-IP packets, the switch uses the data at those fields to compute the hash. IP-based teaming requires that on the ACI side a port-channel / VPC is configured with 'mode on'.	Yes (channel mode set to 'on')
<b>Link Aggregation(LAG) - LACP</b>	LACP Active / Passive	Select an uplink based on a selected hash (20 different hash options available). LACP based teaming requires that on the ACI side a port-channel / VPC is configured with LACP enabled. Make sure to create an Enhanced Lag Policy in ACI and apply it to the VSwitch Policy.	Yes (channel mode set to 'LACP Active/Passive')
<b>Route based on Physical NIC Load (LBT)</b>	MAC Pinning - Physical-NIC-load	Available for distributed port groups or distributed ports. Select an uplink based on the current load of the physical network adapters connected to the port group or port. If an uplink remains busy at 75 percent or higher for 30 seconds, the host's vSwitch moves a part of the virtual machine traffic to a physical adapter that has free capacity.	No

See the screenshot below on how to validate Port-Channel Policy as part of the vSwitch Policy in place.

### ACI vSwitch Policy – Port Channel Policy

Note : For a more in-depth description of VMware networking features, please review vSphere Networking at <https://docs.vmware.com/en/VMware-vSphere/6.5/com.vmware.vsphere.networking.doc/GUID-D34B1ADD-B8A7-43CD-AA7E-2832A0F7EE76.html>

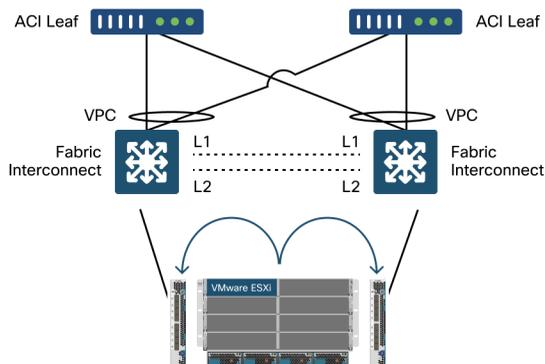
## Cisco UCS B-Series use case

When using Cisco UCS B-Series servers, it is important to note they connect within their chassis to UCS Fabric Interconnects (FIs) that do not have a unified dataplane. This use case equally applies to other vendors which employ a similar topology. Because of

this there can be a difference between the load-balancing method used from an ACI leaf switch side and the vSwitch side.

Below is a UCS FI topology with ACI:

Cisco UCS FI with ACI leaf switches - topology

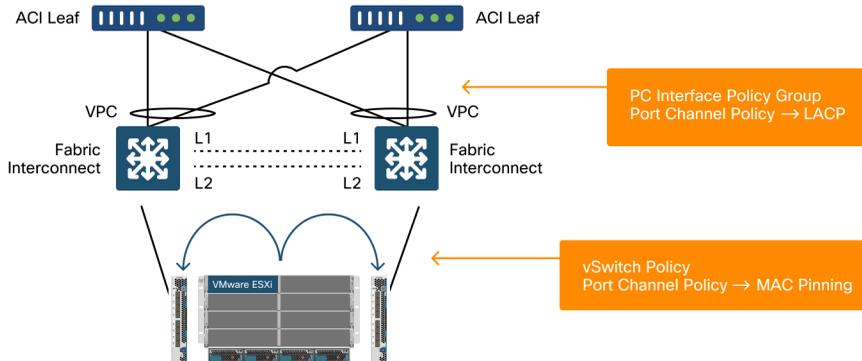


Key things to notice:

- Each Cisco UCS FI has a port-channel towards the ACI leaf switches.
- The UCS FIs are directly interconnected for heartbeat purposes only (not used for dataplane).
- Each blade server's vNIC is pinned to a specific UCS FI or uses a path toward one of FIs by using UCS Fabric Failover (Active-Standby).
- Using IP-hash algorithms on the ESXi host's vSwitch will cause MAC flaps on the UCS FI's.

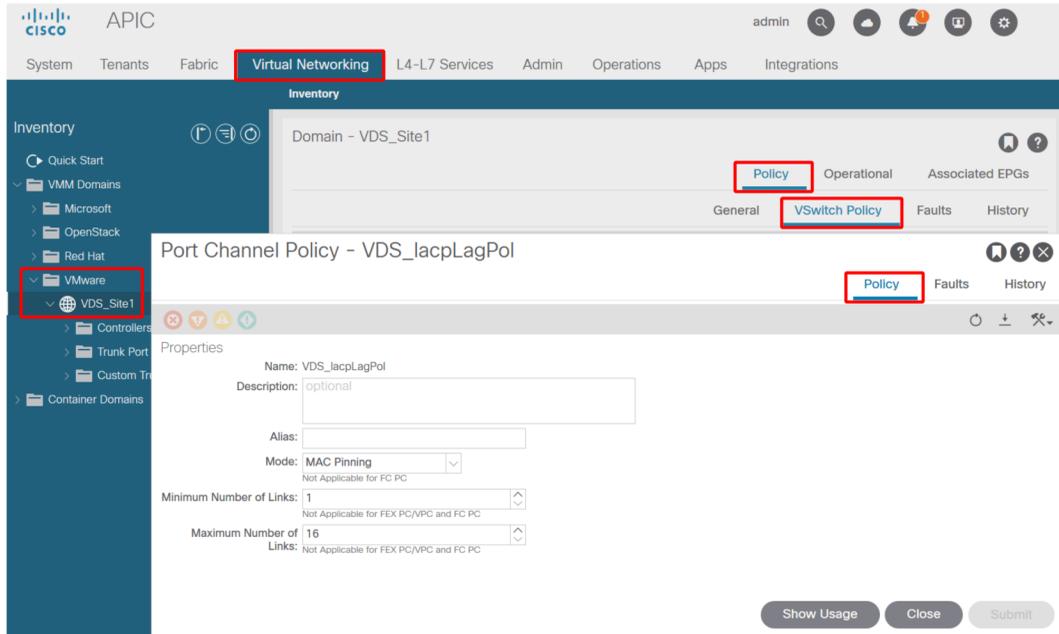
In order to correctly configure this, do the following:

### Cisco UCS FI with ACI leaf switches – port-channel config



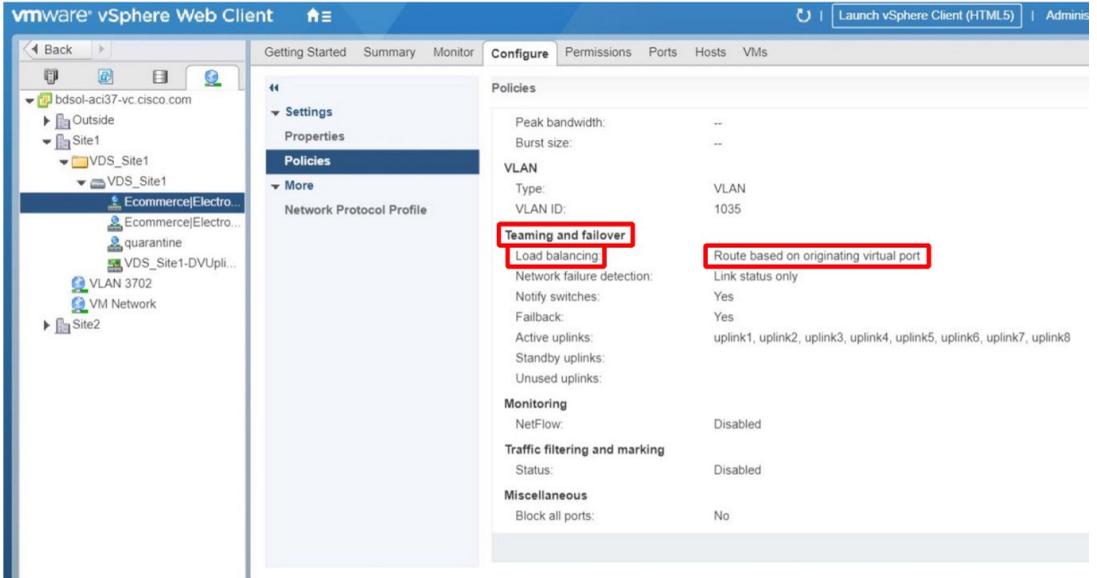
When MAC Pinning is configured on the Port-Channel Policy as part of the vSwitch Policy in ACI, this will show as 'Route based on the originating virtual port' teaming configuration of the port groups on the VDS.

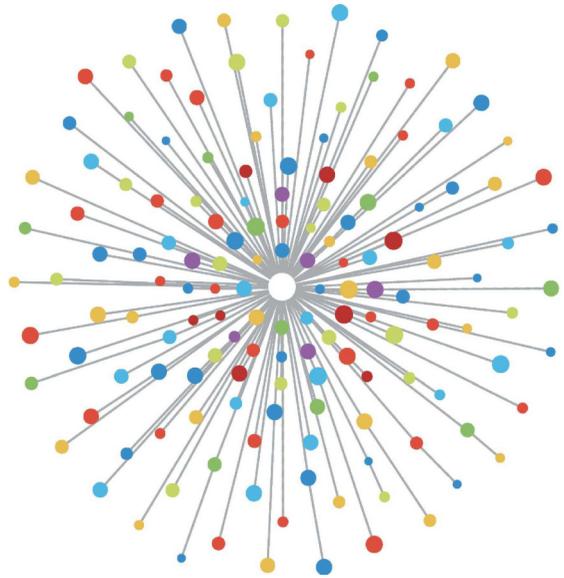
### ACI – Port Channel Policy as part of vSwitch Policy

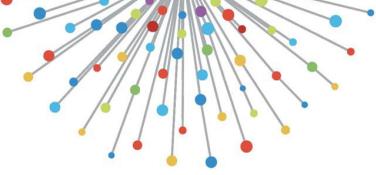


Note: The Port Channel Policy used in the above example is auto-named by the wizard hence it's called "VDS\_lacpLagPol" although we use Mode "MAC Pinning".

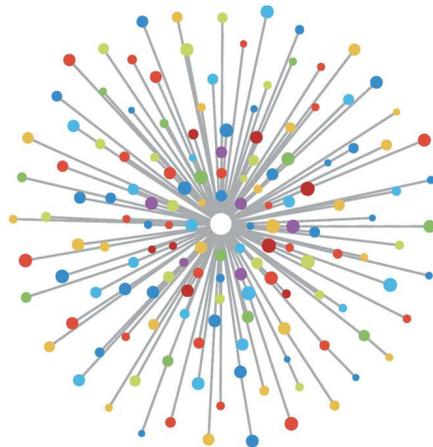
VMWare vCenter – ACI VDS – Port Group – Load Balancing setting







# PBR (Policy-Based Redirect)



## Overview

This chapter explains troubleshooting for unmanaged mode Service Graph with Policy-Based Redirect (PBR).

The following are typical troubleshooting steps. This chapter explains how to verify steps 2 and 3 which are specific to PBR. For steps 1 and 4, please refer to chapters: "Intra-Fabric forwarding", "External forwarding", and "Security policies".

- 1 Check the traffic works without PBR Service Graph:
  - Consumer and provider endpoints are learned.
  - Consumer and provider endpoints can communicate.
- 2 Check Service Graph is deployed:
  - Deployed Graph Instances have no fault.
  - VLANs and class IDs for service node are deployed.
  - Service node endpoints are learned.
- 3 Check the forwarding path:
  - Check policy is programmed on the leaf nodes.
  - Capture the traffic on the service node to confirm if traffic is redirected.
  - Capture the traffic on the ACI leaf to confirm if traffic comes back to the ACI fabric after PBR.
- 4 Check the traffic arrives on the consumer and provider endpoint, and that the endpoint generates the return traffic.

This document doesn't cover design or configuration options. For that information, please refer to the "ACI PBR White Paper" on Cisco.com

In this chapter, service node and service leaf imply the following:

- Service node — an external node to which PBR is redirecting the traffic, such as a firewall or load balancer.
- Service leaf — an ACI leaf that is connected to a service node.

## Service Graph deployment

This chapter explains a troubleshooting example where a Service Graph is not deployed.

After a Service Graph policy is defined and applied to a contract subject, there should be a deployed graph instance appearing on the ACI GUI. The figure below shows the troubleshooting scenario where the Service Graph does not appear as deployed.

Service Graph is not shown as a Deployed Graph Instance.

The screenshot displays the Cisco APIC interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Tenants' section is active, showing 'ALL TENANTS', 'Add Tenant', and a search bar. The tenant 'Prod' is selected, with sub-tenants 'PBR-Multinode' and 'Symmetric-PBR' visible. The left-hand navigation pane is expanded to 'Services' and 'L4-L7', with 'Deployed Graph Instances' highlighted at the bottom. The main content area is titled 'Deployed Graph Instances' and contains a table with the following columns: 'Service Graph', 'Contract', 'Contained By', 'State', and 'Description'. The table is currently empty, displaying the message 'No items have been found.'

### 1. Check configuration steps and fault

The first step of troubleshooting is to check the necessary components have been configured without any fault. The assumption is that general configurations below are already done:

- VRF and BDs for consumer EPG, provider EPG and service node
- The consumer and provider EPG.
- The contract and filters.

It's worth mentioning that an EPG for the service node is not needed to be created manually. It will be created through Service Graph deployment.

Service Graph with PBR configuration steps are the following:

- Create the L4-L7 Device (Logical Device).
- Create the Service Graph.
- Create the PBR policy.
- Create the Device Selection policy.
- Associate the Service Graph with the contract subject.

## 2. Service Graph deployment issues

### Check Service Graph deployment in the UI

After a Service Graph is associated to the contract subject, a deployed graph instance should show up for each contract with Service Graph (figure below).

The location is 'Tenant > Services > L4-L7 > Deployed Graph Instances'

## Deployed Graph Instance

The screenshot displays the Cisco APIC interface. The left sidebar shows the navigation menu with 'Services' and 'L4-L7' folders highlighted in red. Under 'L4-L7', 'Deployed Graph Instances' is also highlighted in red, with a sub-item 'web-to-app-FW-Prod' selected. The main content area shows the 'L4-L7 Service Graph Instance - web-to-app-FW-Prod' configuration. The 'Topology' tab is active, showing a diagram with a central node 'node1' (Prod-ASA...) connected to a 'Consumer' EPG (Web) and a 'Provider' EPG (App). Below the diagram, the 'node1 Information' section lists details such as Contract: Prod/web-to-app, Graph: Prod/FW, Node: node1, Device Cluster: Prod-ASA-VM1, Firewall: routed, and Policy-Based Partition: true. A 'Show Usage' button is located at the bottom right of the information section.

If a Deployed Graph Instance does not show up, there is something wrong with the contract configuration. Major reasons can be:

- The contract doesn't have a consumer or provider EPG.
- The contract subject doesn't have any filter.
- The contract scope is VRF even though it's for inter-VRF or inter-tenant EPG communication.

If Service Graph instantiation fails, faults are raised in the Deployed Graph Instance, which means there is something wrong with the Service Graph configuration. Typical faults caused by configuration are the following:

**F1690: Configuration is invalid due to ID allocation failure**

This fault indicates that the encapsulated VLAN for the service node is not available. For example, there is no available dynamic VLAN in the VLAN pool associated to the VMM domain used in the Logical Device.

Resolution: Check the VLAN pool in the domain used for the Logical Device. Check encapsulated VLAN in the Logical Device interface if it's in a physical domain. The locations are 'Tenant > Services > L4-L7 > Devices and Fabric > Access Policies > Pools > VLAN'.

**F1690: Configuration is invalid due to no device context found for LDev**

This fault indicates that the Logical Device can't be found for the Service Graph rendering. For example, there is no Device Selection Policy matched for the contract with the Service Graph.

Resolution: Check the Device Selection Policy is defined. Device Selection Policy provides a selection criterion for a service device and its connectors. The criteria are based on a contract name, a Service Graph name, and a node name in the Service Graph. The location is 'Tenant > Services > L4-L7 > Device Selection Policy'.

## Check Device Selection Policy

The screenshot shows the Cisco APIC interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', and 'Apps'. The 'Tenants' tab is selected, and the 'Prod' tenant is active. The left navigation menu shows a tree structure under 'Prod', with 'Policies' and 'Devices Selection Policies' highlighted. The main configuration area is titled 'Logical Device Context - web-to-app-FW-node1' and shows the 'Policy' tab. The 'Properties' section includes fields for 'Contract Name: web-to-app', 'Graph Name: FW', and 'Node Name: node1'. The 'Devices' dropdown menu is set to 'Prod-ASAv-VM1'.

**F1690: Configuration is invalid due to no cluster interface found**

This fault indicates that the cluster interface for the service node can't be found. For example, the cluster interface is not specified in Device Selection Policy.

Resolution: Check the cluster interface is specified in Device Selection policy and connector name is correct (Figure below).

**F1690: Configuration is invalid due to no BD found**

This fault indicates that the BD for the service node can't be found. For example, the BD is not specified in Device Selection Policy.

Resolution: Check BD is specified in Device Selection policy and connector name is correct (Figure below).

### F1690: Configuration is invalid due to invalid service redirect policy

This fault indicates that the PBR policy is not selected even though redirect is enabled on the service function in the Service Graph.

Resolution: Select PBR policy in the Device Selection Policy (Figure below).

### Logical interface configuration in Device Selection Policy

The screenshot displays the Cisco APIC interface for configuring a Logical Interface Context. The main content area shows the configuration for the 'consumer' context under the 'Policy' tab. The configuration is as follows:

- Connector Name:** consumer
- Cluster Interface:** consumer
- Associated Network:** Bridge Domain (selected), L3Out
- Bridge Domain:** Service-BD1
- Preferred Contract Group:** Exclude
- Permit Logging:**
- L3 Destination (VIP):**
- L4-L7 Policy-Based Redirect:** ASA-external
- L4-L7 Service EPG Policy:** select an option
- Custom QoS Policy:** select a value
- Subnets:** (empty)

At the bottom of the configuration area, there are three buttons: 'Show Usage', 'Reset', and 'Submit'. The left sidebar shows the navigation tree with 'Services' and 'L4-L7' folders highlighted, and 'Devices Selection Policies' expanded to show 'web-to-app-FW-node1' and 'consumer'.

# Forwarding

This chapter explains the troubleshooting steps for the PBR forwarding path.

## 1. Check VLANs are deployed and endpoints are learned on the leaf node

Once a Service Graph is successfully deployed without any fault, EPGs and BDs for a service node get created. The figure below shows where to find the encapsulated VLAN IDs and class IDs of service node interfaces (Service EPGs). In this example, the consumer side of a firewall is class ID 16386 with VLAN encap 1000 and the provider side of a firewall is class ID 49157 with VLAN encap 1102.

The location is 'Tenant > Services > L4-L7 > Deployed Graph instances > Function Nodes'.

### Service node

The screenshot displays the Cisco APIC interface for configuring a Function Node. The left sidebar shows the navigation tree with 'L4-L7' and 'Service Parameters' highlighted. The main content area shows the 'Function Node - node1' configuration, including 'Cluster Interfaces' and 'Function Connectors' tables.

**Cluster Interfaces:**

Name	Concrete Interfaces	Encap
consumer	Prod-ASAv-VM1.[g9/0]	unknown
provider	Prod-ASAv-VM1.[g9/1]	unknown

**Function Connectors:**

Name	Encap	Class ID
consumer	vlan-1000	16386
provider	vlan-1102	49157

**Meta Folder/Param Key Table:**

Meta Folder/Param Key	Name	Value	Override name/value To

## Service node interface class ID

Function Node - node1

Policy Faults History

Properties

Name: node1  
Function Type: GoTo  
Devices: Prod-ASAv-VM1

Cluster Interfaces:

Name	Concrete Interfaces	Encap
consumer	Prod-ASAv-VM1/g0/0/0	unknown
provider	Prod-ASAv-VM1/g0/0/1	unknown

Function Connectors:

Name	Encap	Class ID
consumer	vlan-1000	16386
provider	vlan-1102	49157

Folders and Parameters

Features

Basic Parameters All Parameters

Meta Folder/Param Key	Name	Value	Override name/value To

These VLANs are deployed on the service leaf node interfaces where the service nodes are connected. VLAN deployment and endpoint learning status can be checked by using 'show vlan extended' and 'show endpoint' on the service leaf node CLI.

```
Pod1-Leaf1# show endpoint vrf Prod:VRF1
Legend:
s - arp          H - vtep          V - vpc-attached  p - peer-aged
R - peer-attached-r1 B - bounce        S - static         M - span
D - bounce-to-proxy O - peer-attached  a - local-aged    m - svc-mgr
L - local        E - shared-service
```

VLAN/ Domain	Encap VLAN	MAC Address IP Address	MAC Info/ IP Info	Interface
53	vlan-1000	0050.56af.3c60	LV	po1
Prod:VRF1	vlan-1000	192.168.101.100	LV	po1
59	vlan-1102	0050.56af.1c44	LV	po1
Prod:VRF1	vlan-1102	192.168.102.100	LV	po1

If endpoint IPs of the service nodes are not learned as endpoints in ACI fabric, it's most likely either a connectivity or configuration issue between the service leaf and service node. Please check the following statuses:

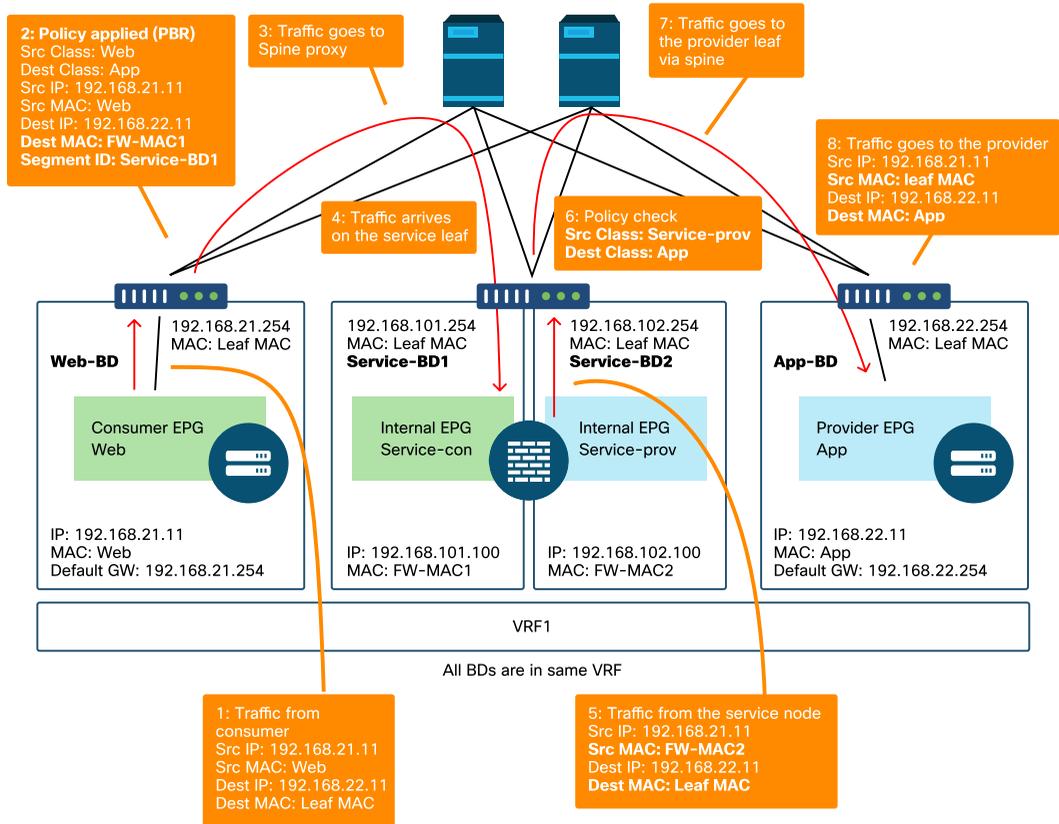
- The service node is connected to the correct leaf downlink port.
  - If the service node is in a physical domain, the leaf static path end encapsulation needs to be defined in the Logical Device.
  - If the service node is in a VMM domain, please check the VMM domain is working and the port group created through Service Graph is attached to the service node VM correctly.
- The leaf downlink port connected to the service node or the hypervisor where the service node VM resides is UP.
- The service node has the correct VLAN and IP address.
- Intermediate switch between the service leaf and the service node has the correct VLAN configuration.

## 2. Check the expected traffic paths

If end-to-end traffic stops working once PBR is enabled, even though the service node endpoints are learned in ACI fabric, the next troubleshooting step is to check what the expected traffic paths are.

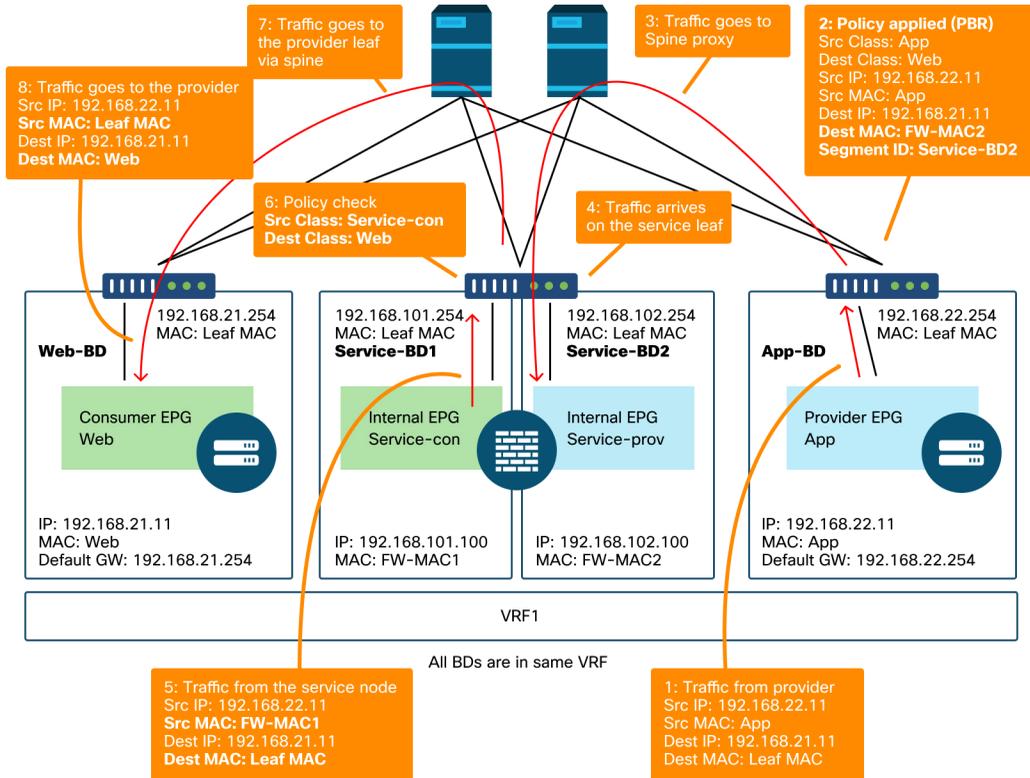
Figures 'PBR forwarding path example - consumer to provider' and 'PBR forwarding path example - provider to consumer' illustrate a forwarding path example of firewall insertion using PBR between a consumer endpoint and a provider endpoint. The assumption is that the endpoints are already learned on leaf nodes.

PBR forwarding path example - consumer to provider



Note : Since source MAC is not changed to ACI leaf MAC, the PBR node must not use source MAC based forwarding if consumer endpoint and PBR node are not in the same BD

PBR forwarding path example - provider to consumer



Note: It's worth mentioning that PBR policy is enforced on either consumer or provider leaf and what ACI PBR does is destination MAC rewrite as shown in figures 'PBR forwarding path example - consumer to provider' and 'PBR forwarding path example - provider to consumer'. Reaching the PBR destination MAC always uses a spine proxy, even if the source endpoint and PBR destination MAC are under the same leaf.

Though figures 'PBR forwarding path example - consumer to provider' and 'PBR forwarding path example - provider to consumer' show an example of where the traffic would be redirected, where policy is enforced depends on contract configuration and endpoint learning status. The table 'Where policy is enforced' summarizes where policy is enforced within a single ACI site. Where policy is enforced in Multi-Site is different.

Where is policy enforced?

Scenario	VRF enforcement mode	Consumer	Provider	Policy enforced on
Intra-VRF	Ingress/egress	EPG	EPG	<ul style="list-style-type: none"> <li>If destination endpoint is learned: ingress leaf*</li> <li>If destination endpoint is not learned: egress leaf</li> </ul>
	Ingress	EPG	L3Out EPG	Consumer leaf (non-border leaf)
	Ingress	L3Out EPG	EPG	Provider leaf (non-border leaf)
	Egress	EPG	L3Out EPG	Border leaf -> non-border leaf traffic <ul style="list-style-type: none"> <li>If destination endpoint is learned: border leaf</li> <li>If destination endpoint is not learned: non-border leaf</li> </ul>
	Egress	L3Out EPG	EPG	Non-border leaf-> border leaf traffic <ul style="list-style-type: none"> <li>Border leaf</li> </ul>
	Ingress/egress	L3Out EPG	L3Out EPG	Ingress leaf*
Inter-VRF	Ingress/egress	EPG	EPG	Consumer leaf
	Ingress/egress	EPG	L3Out EPG	Consumer leaf (Non-border leaf)
	Ingress/egress	L3Out EPG	EPG	Ingress leaf*
	Ingress/egress	L3Out EPG	L3Out EPG	Ingress leaf*

\*Policy enforcement is applied on the first leaf hit by the packet.

These are examples:

- If an external endpoint in L3Out EPG in VRF1 tries to access an endpoint in Web EPG in VRF1 and VRF1 is configured for ingress enforcement mode, traffic is redirected by the leaf where the endpoint in Web EPG resides, regardless of contract direction.
- If an endpoint in consumer Web EPG in VRF1 tries to access an endpoint in provider App EPG in VRF1, and the endpoints are learned on consumer and provider leaf nodes, traffic is redirected by the ingress leaf.
- If an endpoint in consumer Web EPG in VRF1 tries to access an endpoint in provider App EPG in VRF2, traffic is redirected by the consumer leaf where the consumer endpoint resides, regardless of the VRF enforcement mode.

### 3. Check if traffic is redirected to the service node

Once the expected forwarding path is clear, ELAM can be used to check whether traffic arrives on the switch nodes and check the forwarding decision on the switch nodes. Please refer to section "Tools" in the chapter "Intra-Fabric Forwarding" for instructions on how to use ELAM.

For example, to trace the traffic flow in the figure 'PBR forwarding path example - consumer to provider', these can be captured to confirm if consumer to provider traffic is redirected.

- Downlink port on consumer leaf to check 1 and 2 (Traffic arrives on the consumer leaf and PBR is enforced).
- Fabric port on spine nodes to check 3 (Traffic goes to spine proxy).
- Fabric port on service leaf to check 4 (Traffic arrives on the service leaf).

Then, these can be captured to confirm if traffic that comes back from the service node goes to the provider.

- Downlink port on the service leaf to check 5 and 6 (Traffic comes back from the service node and is permitted).

- Fabric port on spine nodes to check 7 (Traffic goes to provider leaf via spine).
- Fabric port on provider leaf to check 8 (Traffic arrives on the service leaf and goes to the provider endpoint).

Note: If consumer and service node are under the same leaf, specify an interface or source MAC in addition to source/destination IP to take ELAM to check 1 or 5 in figure 'PBR forwarding path example - consumer to provider' specifically because both use the same source IP and destination IP.

If the consumer to provider traffic is redirected to the service node but doesn't come back to the service leaf, please check the following as they are common mistakes:

- Service node routing table reaches the provider subnet.
- Service node security policy such as ACL permits the traffic.

If the traffic is redirected and arrives on the provider, please check the return traffic path from provider to consumer in a similar way.

#### 4. Check the policies programmed on leaf nodes

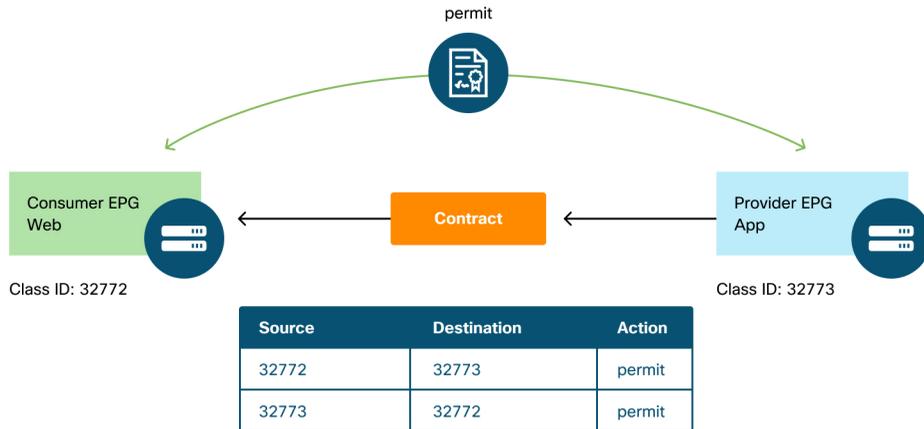
If traffic is not forwarded or redirected accordingly, the next troubleshooting step is to check the policies programmed on the leaf nodes. This section shows zoning-rule and contract\_parser as examples. For more detail of how to check zoning-rules, please refer to section "Tools" in chapter "Security Policies".

Note: The policies are programmed based on EPG deployment status on the leaf. The show command output in this section uses the leaf that has consumer EPG, provider EPG, and EPGs for the service node.

#### Use of the 'show zoning-rule' command

The figure and the 'show zoning-rule' output below describes the zoning-rules before Service Graph deployment.

## Zoning-rules before Service Graph deployment



VRF scope id can be found in 'Tenant > Networking > VRF'.

```
Pod1-Leaf1# show zoning-rule scope 2752513
```

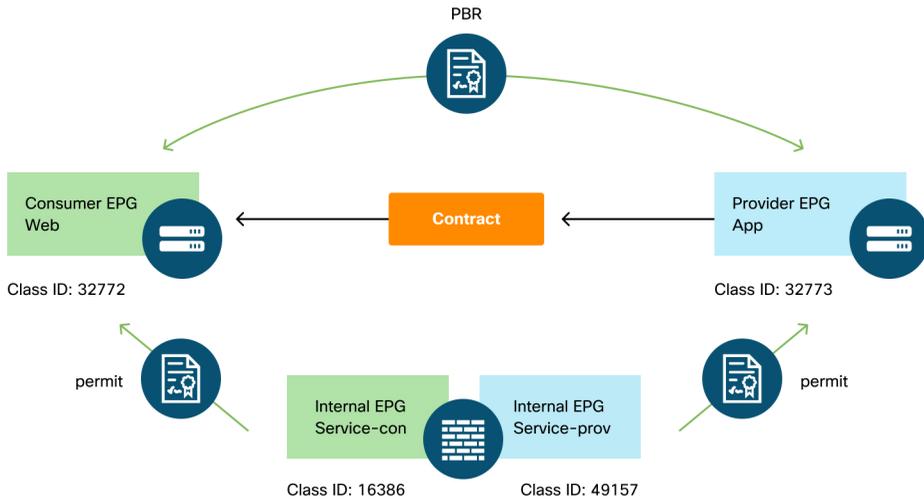
```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir | operSt | Scope | Name | Action | Priority |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 4237 | 32772 | 32773 | 8 | bi-dir | enabled | 2752513 | web-to-app | permit | fully_qual(7) |
| 4172 | 32773 | 32772 | 9 | uni-dir-ignore | enabled | 2752513 | web-to-app | permit | fully_qual(7) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Once the Service Graph is deployed, EPGs for the service node get created and policies are updated to redirect traffic between the consumer and the provider EPGs. The figure below and the 'show zoning-rule' output below describes the zoning-rules after Service Graph deployment. In this example, the traffic from pcTag 32772 (Web) to pcTag 32773 (App) is redirected to 'destgrp-27' (consumer side of the service node) and the traffic from pcTag 32773 (App) to pcTag 32772 (Web) is redirected to 'destgrp-28' (provider side of the service node).

Zoning-rules after Service Graph deployment



Source	Destination	Action
32772	32773	PBR to the consumer side of the service node
49157	32773	permit
32773	32772	PBR to the provider side of the service node
16386	32772	permit

```
Pod1-Leaf1# show zoning-rule scope 2752513
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir | operSt | Scope | Name | Action | Priority |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
...
| 4213 | 16386 | 32772 | 9 | uni-dir | enabled | 2752513 | | permit | fully_qual(7) |
| 4249 | 49157 | 32773 | default | uni-dir | enabled | 2752513 | | permit | src_dst_any(9) |
| 4237 | 32772 | 32773 | 8 | bi-dir | enabled | 2752513 | | redir(destgrp-27) | fully_qual(7) |
| 4172 | 32773 | 32772 | 9 | uni-dir-ignore | enabled | 2752513 | | redir(destgrp-28) | fully_qual(7) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

The destination information of each destgrp can be found by using the 'show service redir info' command.

```

Pod1-Leaf1# show service redir info
=====
LEGEND
TL: Threshold(Low) | TH: Threshold(High) | HP: HashProfile | HG: HealthGrp | BAC: Backup-Dest | TRA: Tracking | RES: Resiliency
=====
List of Dest Groups
GrpID Name          destination          HG-name          BAC operSt   operStQual   TL TH  HP  TRAC RES
=====
28  destgrp-28    dest-[192.168.102.100]-[vxlan-2752513]  Not attached    N  enabled   no-oper-grp   0  0  sym no  no
27  destgrp-27    dest-[192.168.101.100]-[vxlan-2752513]  Not attached    N  enabled   no-oper-grp   0  0  sym no  no

List of destinations
Name                  bdVnid             vMac             vrf             operSt   operStQual   HG-name
=====
dest-[192.168.102.100]-[vxlan-2752513]  vxlan-16023499    00:50:56:AF:1C:44  Prod:VRF1    enabled   no-oper-dest  Not attached
dest-[192.168.101.100]-[vxlan-2752513]  vxlan-16121792    00:50:56:AF:3C:60  Prod:VRF1    enabled   no-oper-dest  Not attached
...

```

If zoning-rules are programmed accordingly, but traffic is not redirected or forwarded accordingly, please check the following as they are common mistakes:

- Check if the source or destination class ID is resolved as expected by using ELAM. If not, please check what the wrong class ID is and the EPG derivation criteria such as path and encap VLAN.
- Even though source and destination class IDs are resolved accordingly, and PBR policy is applied but traffic doesn't arrive on the PBR node, please check IP, MAC, and VRF of the destgrp in the redir action ('show service redir info') are correct.

By default, permit rules for a consumer EPG to a service node (consumer side), and a provider EPG to a service node (provider side) are not programmed if PBR is enabled. Thus, a consumer or provider endpoint can't directly communicate to the service node by default. To permit this traffic, the Direct Connect option needs to be enabled. The use case is explained in section "Other traffic flow examples".

### Use of `contract_parser`

The `contract_parser` tool can also help to verify the policies. C-consumer is the consumer side of the service node and C-provider is the provider side of the service node.

```
Pod1-Leaf1# contract_parser.py --vrf Prod:VRF1
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-l4] dst-epg [dst-l4] [flags][contract:{str}] [hit-count]

[7:4213] [vrf:Prod:VRF1] permit ip tcp tn-Prod/G-Prod-ASAv-VM1ctxVRF1/C-consumer(16386) eq 80 tn-Prod/ap-
app1/epg-Web(32772) [contract:uni/tn-Prod/brc-web-to-app] [hit=0]
[7:4237] [vrf:Prod:VRF1] redir ip tcp tn-Prod/ap-app1/epg-Web(32772) tn-Prod/ap-app1/epg-App(32773) eq 80
[contract:uni/tn-Prod/brc-web-to-app] [hit=0]
                                destgrp-27 vrf:Prod:VRF1 ip:192.168.101.100 mac:00:50:56:AF:3C:60 bd:uni/tn-
Prod/BD-Service-BD1
[7:4172] [vrf:Prod:VRF1] redir ip tcp tn-Prod/ap-app1/epg-App(32773) eq 80 tn-Prod/ap-app1/epg-Web(32772)
[contract:uni/tn-Prod/brc-web-to-app] [hit=0]
                                destgrp-28 vrf:Prod:VRF1 ip:192.168.102.100 mac:00:50:56:AF:1C:44 bd:uni/tn-
Prod/BD-Service-BD2
[9:4249] [vrf:Prod:VRF1] permit any tn-Prod/G-Prod-ASAv-VM1ctxVRF1/C-provider(49157) tn-Prod/ap-app1/epg-
App(32773) [contract:uni/tn-Prod/brc-web-to-app] [hit=15]
...
```

## Other traffic flow examples

This section considers other common traffic flow examples to identify the desired flows for troubleshooting. For troubleshooting steps, please refer to the previous chapter in this section.

### 1 Load balancer without SNAT:

- In this example, consumer EPG Web and provider EPG App have a contract with a load balancer Service Graph. Endpoints in App EPG are real servers associated to the VIP on the load balancer.
- PBR to load balancer is enabled for provider to consumer traffic direction.

### 2 Firewall and load balancer without SNAT:

- In this example, consumer EPG Web and provider EPG App have a contract with a firewall and a load balancer Service Graph. Endpoints in App EPG are real servers associated with the VIP on load balancer.
- PBR to firewall is enabled for both directions.
- PBR to load balancer is enabled for provider to consumer traffic direction.

### 3 Shared service (Inter-VRF contract):

- In this example, consumer EPG Web and provider EPG App have a contract with a firewall Service Graph. EPG Web and EPG App are in different VRFs.
- PBR to firewall is enabled for both directions.
- The firewall is in between VRFs.

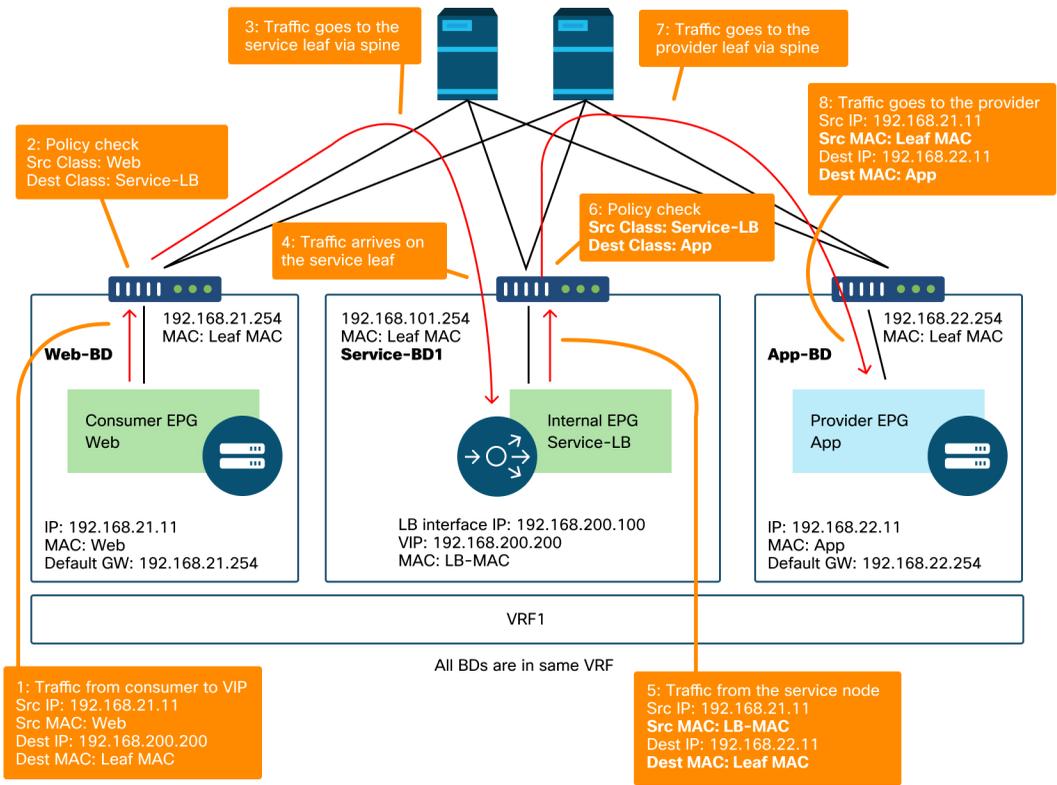
### 1. Load balancer without SNAT

PBR can be deployed as bidirectional PBR or unidirectional PBR. One use case for unidirectional PBR is load balancer integration without source Network Address Translation (NAT). If load balancer performs source NAT, PBR is not required.

#### Traffic path example

The figure below illustrates an example of an incoming traffic flow from consumer EPG Web to provider EPG App with two connections: One is from an endpoint in the consumer EPG Web to the load balancer VIP, and the other is from the load balancer to an endpoint in the provider EPG App. Because the incoming traffic is destined to the VIP, the traffic will reach the load balancer without PBR if the VIP is reachable. The load balancer changes the destination IP to one of the endpoints in EPG App associated to the VIP but doesn't translate the source IP. Accordingly, traffic goes to the provider endpoint.

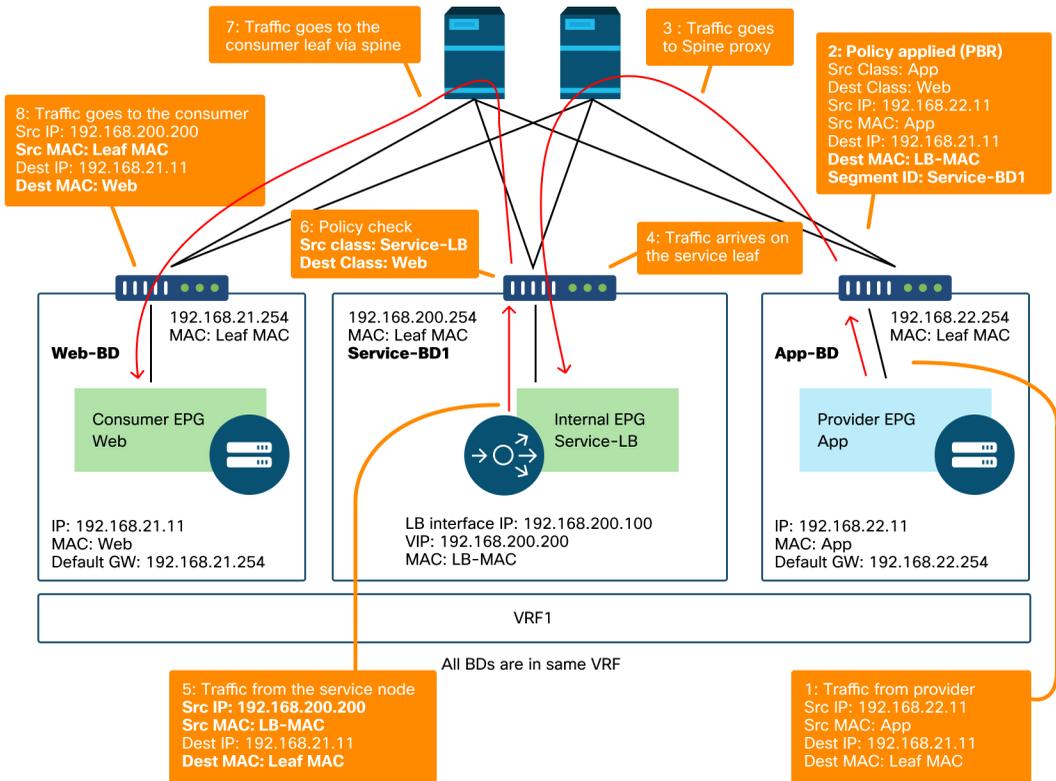
Load balancer without SNAT forwarding path example – consumer to VIP and load balancer to provider without PBR



The figure below illustrates the return traffic flow from provider EPG App to consumer EPG Web. Because the return traffic is destined to the original source IP, PBR is required to make the return traffic to go back to the load balancer. Otherwise the consumer endpoint receives the traffic where the source IP is the provider endpoint instead of the VIP. Such traffic will be dropped because the consumer endpoint didn't initiate traffic to the provider endpoint even if the intermediate network such as the ACI fabric forwards the packet back to the consumer endpoint.

After the traffic from the provider endpoint to the consumer endpoint is redirected to the load balancer, the load balancer changes the source IP to the VIP. Then, the traffic comes back from the load balancer and the traffic goes back to the consumer endpoint.

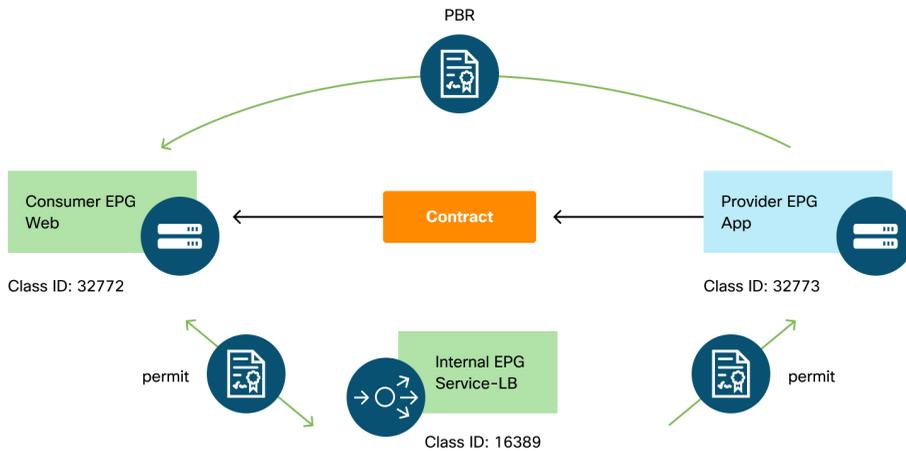
Load balancer without SNAT forwarding path example - provider to consumer with PBR



**The policies programmed on the leaf nodes.**

The figure below and the 'show zoning-rule' output below describe the zoning-rules after Service Graph deployment. In this example, the traffic from pcTag 32772 (Web) to pcTag 16389 (Service-LB) is permitted, the traffic from pcTag 16389 (Service-LB) to pcTag 32773 (App) is permitted, and the traffic from pcTag 32773 (App) to pcTag 32772 (Web) is redirected to 'destgrp-31' (load balancer).

Zoning-rules after Service Graph deployment - load balancer without SNAT



Source	Destination	Action
32772	16389	permit
16389	32773	permit
32773	32772	PBR to the service node
16389	32772	permit

```
Pod1-Leaf1# show zoning-rule scope 2752513
```

Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name	Action	Priority
4248	16389	32773	default	uni-dir	enabled	2752513		permit	src_dst_any(9)
4143	32773	32772	9	uni-dir	enabled	2752513		redir(destgrp-31)	fully_qual(7)
4234	16389	32772	9	uni-dir-ignore	enabled	2752513		permit	fully_qual(7)
4133	32772	16389	8	bi-dir	enabled	2752513		permit	fully_qual(7)

By default, a permit rule for provider EPG (pcTag 32773) to Service-LB (pcTag 16389) is not programmed. To permit bi-directional communication between them for health-checks from the load balancer to provider endpoints, the Direct Connect option on the connection must be set to True. The location is 'Tenant > L4-L7 > Service Graph Templates > Policy'. The default value is False.

### Set Direct Connect option

The screenshot shows the Cisco APIC interface. The navigation menu on the left is expanded to 'Services > L4-L7 > Service Graph Templates > LB'. The main content area displays the 'L4-L7 Service Graph Template - LB' configuration page. The 'Policy' tab is selected, showing a table of connections. The 'Direct Connect' column for connection C2 is highlighted with a red box and set to 'True'. An orange callout box points to this value with the text: 'C2 is the connection between provider EPG and provider side of service node'.

Name	Connected Nodes	Direct Connect	Unicast Route	Adjacency Type	Description
C1	N1, T1	False	True	L3	
C2	N1, T2	True	True	L3	

It adds a permit rule for provider EPG(32773) to Service-LB(16389) as below.

```
Pod1-Leaf1# show zoning-rule scope 2752513
```

Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name	Action	Priority
4248	16389	32773	default	bi-dir	enabled	2752513		permit	src_dst_any(9)
4143	32773	32772	9	uni-dir	enabled	2752513		redir(destgrp-31)	fully_qual(7)
4234	16389	32772	9	uni-dir-ignore	enabled	2752513		permit	fully_qual(7)
4133	32772	16389	8	bi-dir	enabled	2752513		permit	fully_qual(7)
<b>4214</b>	<b>32773</b>	<b>16389</b>	<b>default</b>	<b>uni-dir-ignore</b>	<b>enabled</b>	<b>2752513</b>		<b>permit</b>	<b>src_dst_any(9)</b>

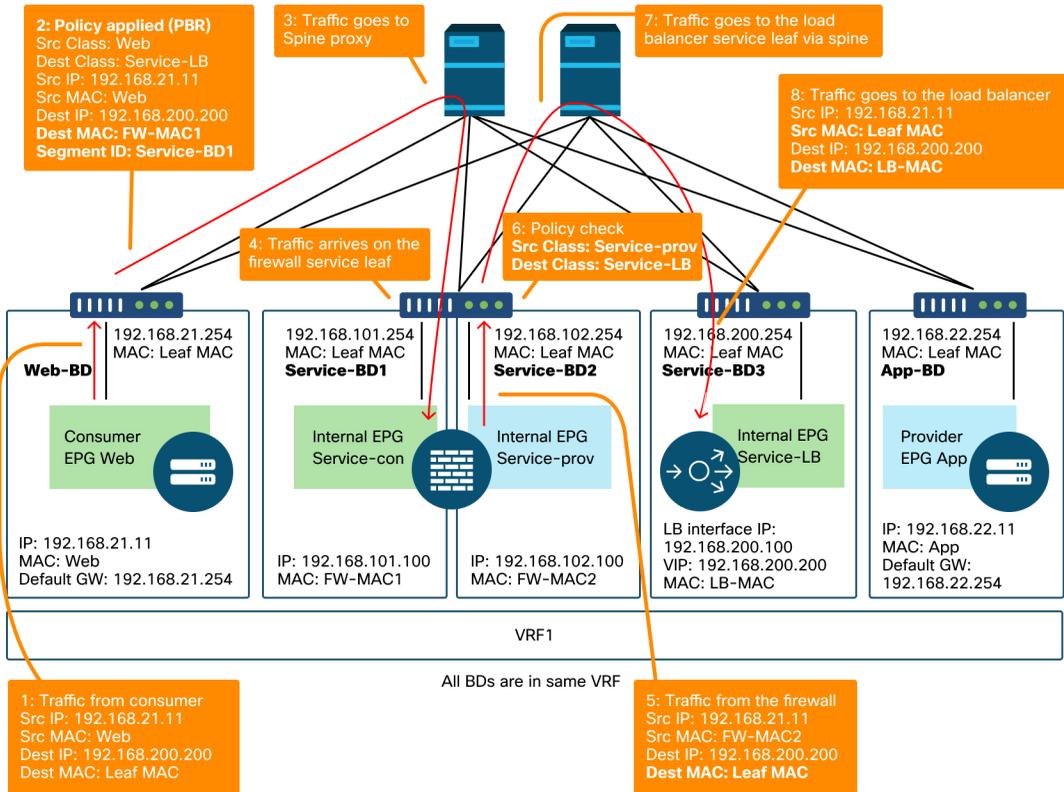
## 2. Traffic flow example - Firewall and load balancer without SNAT

PBR can be deployed with multiple service functions in a Service Graph such as firewall as first node and load balancer as second node.

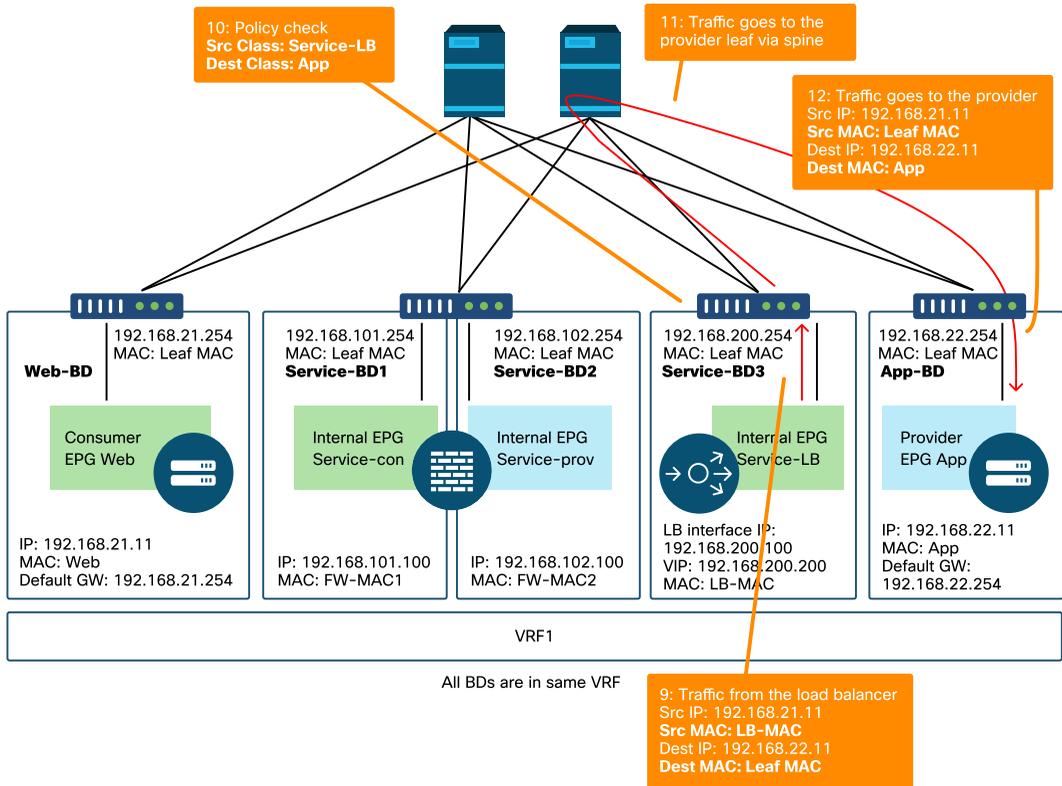
### Traffic path example

The figure below illustrates an example of an incoming traffic flow from consumer EPG Web to provider EPG App with two connections: One is from an endpoint in the consumer EPG Web to the load balancer VIP via firewall and the other is from the load balancer to an endpoint in the provider EPG App. The incoming traffic destined to the VIP is redirected to the firewall and then goes to the load balancer without PBR. The load balancer changes the destination IP to one of the endpoints in App EPG associated to the VIP but doesn't translate the source IP. Then, traffic goes to the provider endpoint.

Firewall and load balancer without SNAT forwarding path example - consumer to VIP and load balancer to provider



Firewall and load balancer without SNAT forwarding path example - consumer to VIP and load balancer to provider (continued)

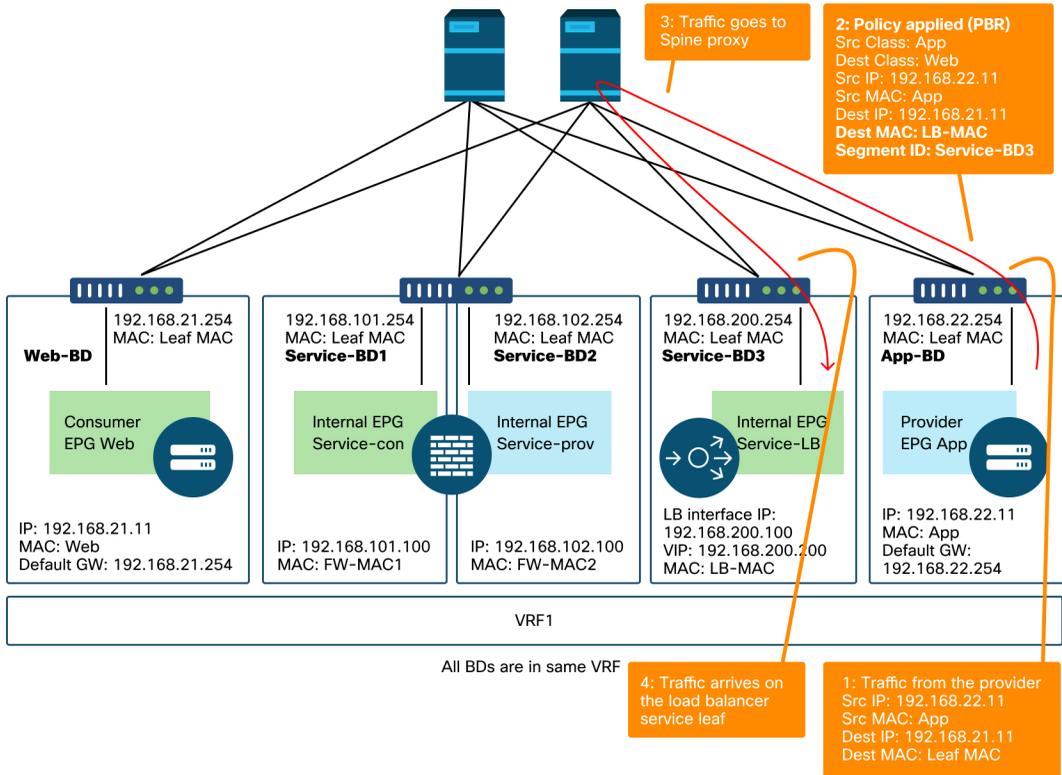


The figure below illustrates the return traffic flow from provider EPG App to consumer EPG Web. Because the return traffic is destined to original source IP, PBR is required to make the return traffic go back to the load balancer.

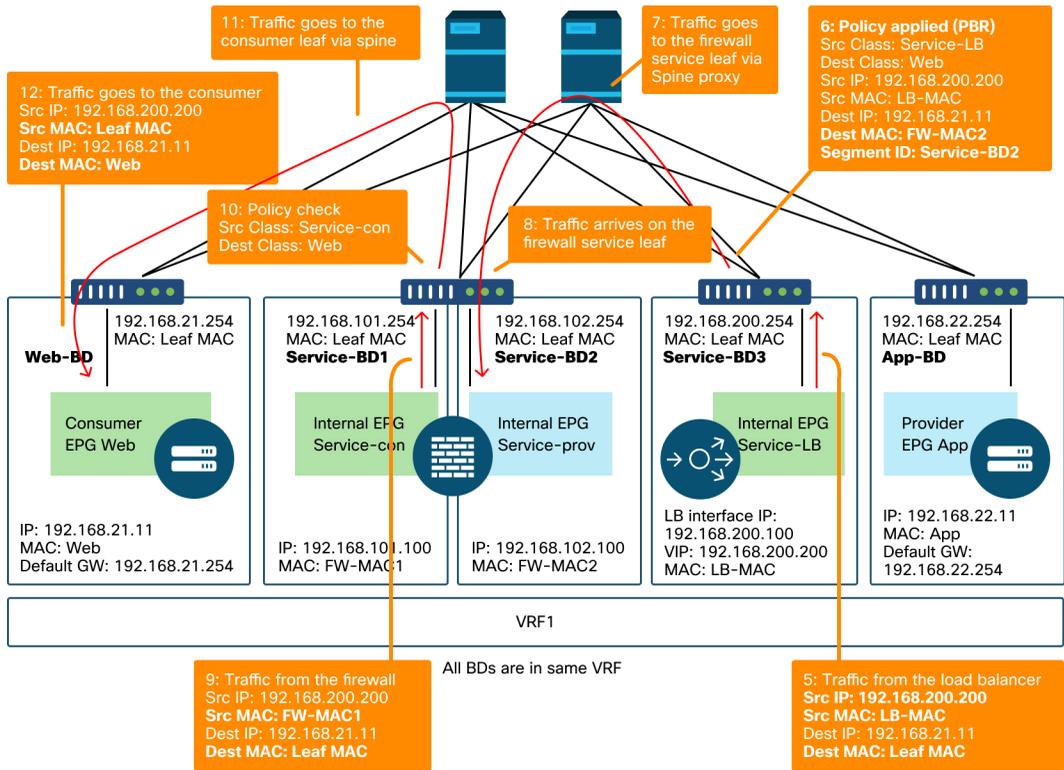
After the traffic from the provider endpoint to the consumer endpoint is redirected to the load balancer, the load balancer changes the source IP to the VIP. The traffic comes

back from the load balancer and is redirected to the firewall. Then, the traffic comes back from the firewall and goes back to the consumer endpoint.

Firewall and load balancer without SNAT forwarding path example - provider to consumer



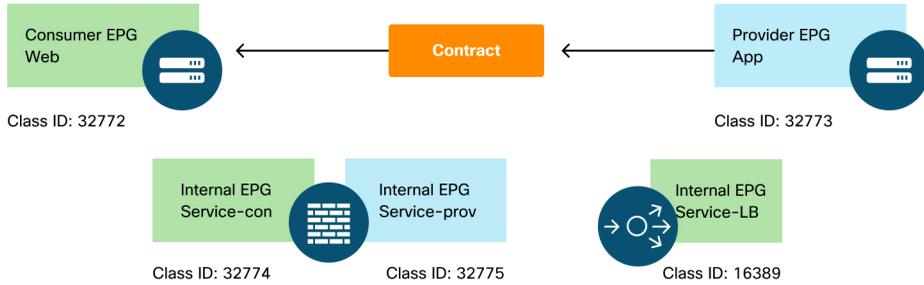
## Firewall and load balancer without SNAT forwarding path example - provider to consumer (continued)



### The policies programmed on the leaf nodes

The figure below and the 'show zoning-rule' output shown below describe the zoning-rules after Service Graph deployment. In this example, the traffic from pcTag 32772 (Web) to pcTag 16389 (Service-LB) is redirected to 'destgrp-32' (consumer side of the firewall), the traffic from pcTag 32773 (App) to pcTag 32772 (Web) is redirected to 'destgrp-33' (load balancer), and the traffic from pcTag 16389 (Service-LB) to pcTag 32772 (Web) is redirected to 'destgrp-34' (provider side of the firewall).

Zoning-rules after Service Graph deployment - firewall and load balancer without SNAT



Source	Destination	Action
32772	16389	PBR to the consumer side of the firewall
32775	16389	permit
16389	32773	permit
32773	16389	Permit (Direct Connect must be set to True)
32773	32772	PBR to the the load balancer
16389	32772	PBR to the provider side of the firewall
32774	32772	permit

```
Pod1-Leaf1# show zoning-rule scope 2752513
```

Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name	Action	Priority
4236	32772	16389	8	bi-dir	enabled	2752513	redir(destgrp-32)	fully_qual(7)	
4143	32773	32772	9	uni-dir	enabled	2752513	redir(destgrp-33)	fully_qual(7)	
4171	16389	32773	default	bi-dir	enabled	2752513	permit	src_dst_any(9)	
4248	16389	32772	9	uni-dir-ignore	enabled	2752513	redir(destgrp-34)	fully_qual(7)	
4214	32774	32772	9	uni-dir	enabled	2752513	permit	fully_qual(7)	
4244	32775	16389	default	uni-dir	enabled	2752513	permit	src_dst_any(9)	
4153	32773	16389	default	uni-dir-ignore	enabled	2752513	permit	src_dst_any(9)	

In the example above, the Direct Connect option is set to 'True' on the connection between the provider side of the load balancer and the provider EPG. It must be enabled for health-check from the load balancer to provider endpoints. The location is 'Tenant > L4-L7 > Service Graph Templates >Policy'. Please refer to figure 'Set Direct Connect option'.

### 3. Shared service (Inter-VRF contract)

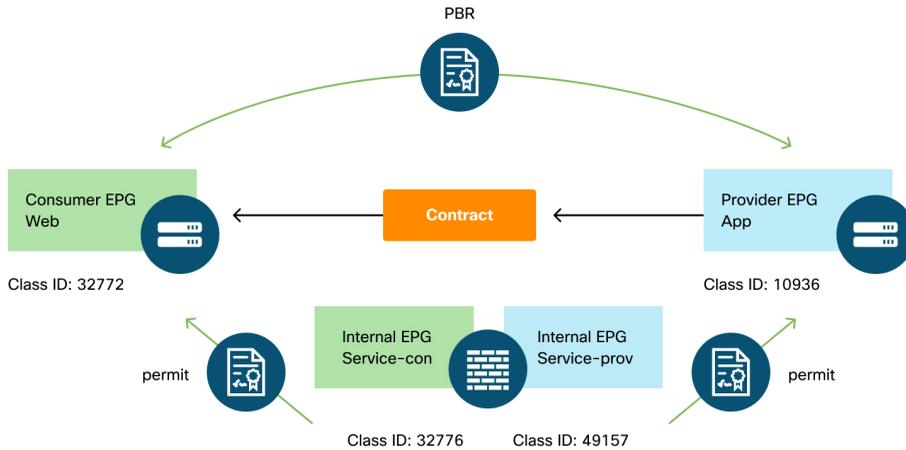
PBR can be enabled in inter-VRF contract. This section explains how the zoning-rules are programmed in the case of EPG to EPG inter-VRF contract.

#### The policies programmed on the leaf nodes

In case of EPG to EPG inter-VRF contract, policy is always enforced in consumer VRF. Thus, redirection happens on the consumer VRF. For other combinations, please refer to table "Where is policy enforced?" in section "Forwarding".

The figure below and the 'show zoning-rule' output below describes the zoning-rules after Service Graph deployment. In this example, the traffic from pcTag 32772 (Web) to pcTag 10936 (App) is redirected to 'destgrp-36' (consumer side of the service node) and the traffic from pcTag 10936 (App) to pcTag 32772 (Web) is redirected to 'destgrp-35' (provider side of the service node). Both are enforced in VRF1 that is consumer VRF. The traffic from pcTag 32776 (consumer side of the firewall) to pcTag 32772 (Web) is permitted in VRF1.

Zoning-rules after Service Graph deployment - inter-VRF contract



**VRF1 (scope: 2752513)**

VRF1 zoning-rules

Source	Destination	Action
32772	10936	PBR to the consumer side of the service node
10936	32772	PBR to the provider side of the service node
32776	32772	permit

**VRF2 (scope: 2555904)**

VRF2 zoning-rule

Source	Destination	Action
49157	10936	permit

Pod1-Leaf1# show zoning-rule scope 2752513

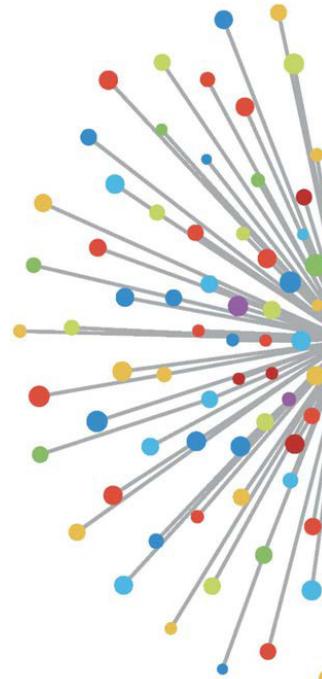
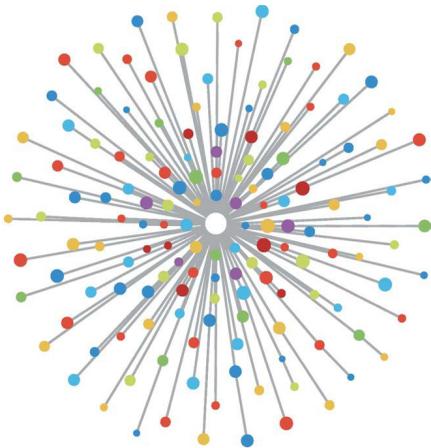
Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name	Action	Priority
4191	32776	32772	9	uni-dir	enabled	2752513		permit	fully_qual(7)
4143	10936	32772	9	uni-dir-ignore	enabled	2752513		redir(destgrp-35)	fully_qual(7)
4136	32772	10936	8	bi-dir	enabled	2752513		redir(destgrp-36)	fully_qual(7)

The traffic from pcTag 49157 (provider side of the firewall) to pcTag 10936 (App) is permitted in VRF2 because both are in VRF2.

```
Pod1-Leaf1# show zoning-rule scope 2555904
```

Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name	Action	Priority
4249	49157	10936	default	uni-dir	enabled	2555904		permit	src_dst_any(9)

# Fabric upgrade



## Overview

The ACI fabric allows for upgrade of a multitude of systems (APICs, leaf nodes, and spines) using a well-defined workflow.

The specific steps and verifications that need to be performed before, during, and after the upgrade process are the focus of discussion in this chapter. The recommendations and guidelines outlined are of utmost consideration with respect to ensuring maximum availability throughout the process and ultimately the success of the upgrade procedure.

This section will also cover specific areas such as APIC CIMC and Switch EPLD/FPGA/BIOS upgrade scenarios.

Cisco's recommendations related to the upgrade procedure focus on preparing an upgrade as per the official guidelines, release notes, advisories, and documents available on Cisco.com. If the upgrade should fail for any reason, basic troubleshooting scenarios will be advised in this document. To limit impact of a failed upgrade, Cisco's recommendation is to contact Cisco TAC for expert guidance and assistance.

# Pre-upgrade validations

When upgrading an ACI fabric, proper planning and preparation is strongly advised. To help facilitate with these preparations, Cisco has multiple documents and tools available that can be leveraged to ensure a predictable and reliable upgrade. Additional consideration is given to potential fallback plans in the event of an unexpected outcome.

## Verify cluster health

The health of the APIC cluster should be validated prior to fabric upgrade. If any issues or anomalies are observed, further troubleshooting and resolution should take place before proceeding.

```
admin@bdsol-aci12-apic1:~> acdiag cluster
Admin password:
Product-name = APIC-SERVER-M1
Serial-number = FCH1906V1XV
Running...

Checking Core Generation: OK
Checking Wiring and UUID: OK
Checking AD Processes: Running
Checking All Apics in Commission State: OK
Checking All Apics in Active State: OK
Checking Fabric Nodes: OK
Checking Apic Fully-Fit: OK
Checking Shard Convergence: OK
Checking Leadership Degration: Optimal leader for all shards
Ping OOB IPs:
APIC-1: 10.48.22.69 - OK
APIC-2: 10.48.22.70 - OK
APIC-3: 10.48.22.71 - OK
Ping Infra IPs:
APIC-1: 10.0.0.1 - OK
APIC-2: 10.0.0.2 - OK
APIC-3: 10.0.0.3 - OK
Checking APIC Versions: Same (4.2(1c))
Checking SSL: OK

Done!
```

## Upgrade checklist

The starting point for ACI upgrade preparations should always be the 'Cisco ACI Upgrade Checklist' found on Cisco.com.

Additional important recommendations are listed below:

- Check the **Release Notes** for latest information regarding the target software version.
- Check for **Faults** and remediate where possible before starting the upgrade.
- Review **Hardware Support Information** for ACI-mode switches documented in the "ACI-mode Switches Hardware Support Matrix".
- Determine the appropriate and supported upgrade path using the "APIC Upgrade/Downgrade Support Matrix".
- Review ACI upgrade best practices:
  - Understand how the **encryption backup** and **passphrase** works.
  - Disable any **apps** installed on the Cisco APIC nodes before upgrading the APIC software on those nodes.
  - Verify that the **Target Version** is set correctly before performing an upgrade.
  - Confirm that the '/firmware' partition is not filled beyond 75%.
- Use **StateChecker** available on Cisco DC App Center (also known as StateChangeChecker).
- Review "Installation Notes and Usage Guidelines" documents.
- Understand **firmware management** and review the **Important Notes** in the "Cisco APIC Installation, Upgrade, and Downgrade Guide".

- Understand firmware upgrade modes (e.g. 'Upgrade now' and 'Schedule an upgrade for later').
- Understand the workflow to upgrade the Cisco ACI fabric documented in the "Cisco APIC Installation, Upgrade, and Downgrade Guide".
- Understand upgrading the Cisco APIC node and the switch software documented in the "Cisco APIC Installation, Upgrade, and Downgrade Guide".
- Upgrade the Cisco APIC and switches software documented in the "Cisco APIC Installation, Upgrade, and Downgrade Guide".

Make sure to review these checks in detail. What follows is a deep dive into the most significant steps.

### **Check Release Notes for late-breaking information.**

Cisco cannot emphasize enough the importance of Release Notes. These notes specify which hardware is supported, which caveats are to be reviewed, which specific considerations have to be taken into account, as well as other version specific information. Make sure to properly review and read the Release Notes for the target version and if there are any concerns, reach out to Cisco TAC for further assistance. Please do know there are Release Notes for both the Switch and APIC software.

### **Check for faults before going through the upgrade**

Before upgrading, be sure to review the faults in the system. As per version 4.2, the fabric will refuse to upgrade when certain critical or major faults are active in the system, unless the check is acknowledged and overridden. Please review the section "Blocking ACI Upgrades or Downgrades If Faults Are Present" in the Cisco APIC Installation, Upgrade, and Downgrade Guide. In summary, all critical faults and some major faults will block the upgrade.

The upgrade dialog will show the following when launching an upgrade in a fabric with critical or targeted major faults.

### Warning for critical faults

Schedule Controller Upgrade 🔊 ? ✕

✕ Migration cannot proceed due to 1 active critical config faults. It's recommended that these faults are resolved before performing a controller upgrade. All unsupported features must be disabled before downgrade to avoid unpredictable behavior. [Click Here](#) for more info.

I understand there are active faults on the system which can lead to unexpected issues, proceed with the upgrade.

Target Firmware Version:  🔄

Upgrade Start Time:

Ignore Compatibility Check:

The administrator has the option to acknowledge these faults by selecting the checkbox pictured above. Do understand that proceeding might have severe impact on the health of the fabric and predictability of a successful upgrade.

### Determine the appropriate upgrade path.

Refer to the 'APIC Upgrade/Downgrade Support Matrix' tool to verify a supported upgrade path and to review which caveats should be taken into consideration.

## APIC Upgrade/Downgrade Support Matrix



This page provides Cisco APIC software upgrade and downgrade information based on current and target releases. The provided upgrade paths have been tested and validated by Cisco, Cisco partners, or both.

For an overview of the entire fabric upgrade process, including relevant reference and procedure documents, see the [Cisco ACI Upgrade Checklist](#).

For feedback on this tool, send email to [apic-docfeedback@cisco.com](mailto:apic-docfeedback@cisco.com).

I am upgrading...  I am downgrading...

From release

To release

**Current release:** 3.2(1)

**Target release:** 4.2(1) [[↗](#)]

**Recommended path:** 3.2(1) → 3.2(7) → 4.2(1) [[Show All](#)]

**Procedure:**

- Upgrade the Cisco APICs. Unless otherwise stated, we recommend upgrading to the latest letter release in the target release train.
- After the Cisco APICs are upgraded successfully, upgrade the switches using 2 or more maintenance groups.
- After the APICs and the switches are upgraded successfully, upgrade the Cisco ACI Virtual Edge or Cisco AVS.

When using the tool, be sure to scroll down to the bottom section and review the 'Target Release - Open Bugs' and "Current Release - Bug Status" to understand the impact of the proposed upgrade path.

The tool will provide all possible upgrade paths if multiple paths are available. It is strongly advised to use the paths going over the 'Long-Lived' APIC versions such as 3.2(x) and 4.2(x).

Refer to the 'Recommended Cisco APIC' and "Cisco Nexus 9000 Series ACI-Mode Switches Releases" documents regarding Long-Lived Releases.

### Understand how the encryption backup and passphrase works.

APIC backups do not use encryption by default. To use encryption, a user must manually enable this feature and set the specific key value that is used for encryption and decryption. Once defined, it is not possible to perform a key lookup. This makes it vital to have the key saved elsewhere and readily available in order to do a restore of the backup should any issues arise.

Should an issue arise during the fabric upgrade that makes it necessary to restore from backup, be sure to contact Cisco TAC for assistance in confirming and validating next steps.

### Use the StateChecker application

There is an app in ACI appcenter called 'State Change Checker' that can capture the state of the fabric before and after an upgrade to assist in validating any changes incurred to the state of the fabric as a result of the upgrade.

This functionality proves very useful in the following circumstances:

- Validation that after reboot, a leaf or spine, important configuration has not gone missing.
  - There may, for example, be missing routes after reboot due to an external device responding on the reboot event.
- Overlapping policies which are applied in a different order after reboot.

Evaluating change purely based on number of objects is not a foolproof method in understanding whether something has gone wrong during upgrade. There is always a possibility that the number of objects will change across versions. However, it is worth referencing this tool in the event that impact is felt, to understand if the tool can help in narrowing down the investigation.

The following is an example of a spine where a snapshot was taken before and after upgrade. In the example, OSPF routes are observed to have gone missing.

## State Change Checker view

Comparison details for Spine201 - Initial ↔ Spine201 - Upgraded

Definition Full

623	8	40	309
Equal	Created	Modified	Deleted

Class view  Node view  Include empty results

Search

Class	Equal	Created	Modified	Deleted	Actions
arpAdjEp	0	0	0	2	
bgpDom	1	0	1	0	

The specific objects deleted can be observed by drilling down on the red text.

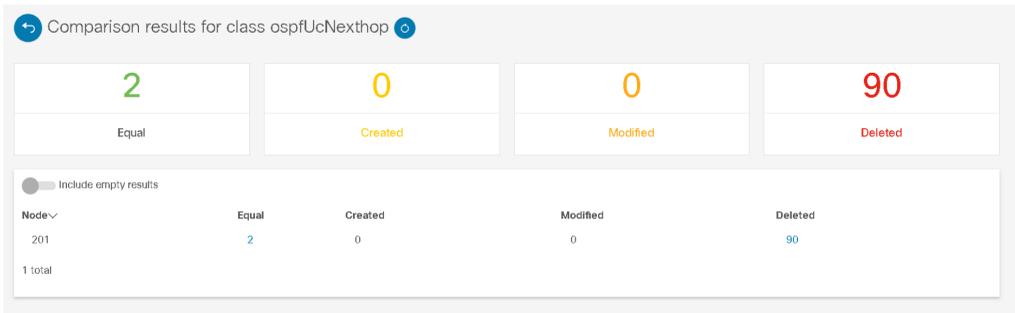
## State Change Checker drill down into 'Deleted' section

Class	Equal	Created	Modified	Deleted	Actions
isisOflIstSpine	2	0	15	3	
isisRoute	13	0	0	11	
lldpAdjEp	2	0	0	2	
lldpIfl	32	0	2	0	
ospfAdjEp	0	0	0	2	
ospfIfl	2	0	2	0	
ospfLsaRec	9	0	1	0	
ospfRoute	2	0	0	54	
ospfUcNextHop	2	0	0	90	
tunnellf	6	0	4	1	

22 total ◀ 1 2 3 ▶

Another level of detail can be achieved by drilling down on the ospfRoute object.

## State Change Checker detail ospfUcNexthop class



Drilling down further, it is observed which node specifically incurred the deletions.

## State Change Checker detail ospfUcNexthop class



Further detail can illustrate which specific OSPF next-hops were deleted.

## Follow the recommended upgrade procedure

When upgrading an ACI fabric, make sure to follow the recommended procedure as outlined in the "Cisco APIC Installation, Upgrade, and Downgrade" guide.

At a high-level, the following are the steps to upgrade a Cisco ACI fabric:

- The procedure or steps for upgrade and downgrade are the same unless stated otherwise in the Release Notes of a specific release.
- Ensure that the CIMC software is a version that interoperates with the destination APIC software.
- Download the Cisco ACI Controller image (Cisco APIC image) to the APIC firmware repository.
- Download the Cisco ACI switch image to the APIC firmware repository.
- Upgrade the cluster of Application Policy Infrastructure Controllers (Cisco APICs).
- Verify that the fabric is operational and the APIC cluster is 'Fully Fit' before proceeding.
- Divide the switches into multiple groups, and upgrade the switches by group, verifying that the fabric is operational between switch group upgrades. For example, divide the switches into two groups - red and blue, then proceed with the following upgrade process:
  - 1 Upgrade the red group of switches.
  - 2 Wait for the red group of switches to finish upgrading.
  - 3 Verify that the fabric is operational.
  - 4 Upgrade the blue group of switches.
  - 5 Wait for the blue group of switches to finish upgrading.
  - 6 Verify that the fabric is operational.

For CIMC upgrade, refer to the "CIMC" section in this chapter.

Divide switches into two or more groups depending on scale of the fabric.  
Depending on scale, some fabrics may require multiple firmware upgrade groups.  
Upgrade one group at a time. That way, fabric bandwidth will not be lost entirely during the upgrade window.

# During and POST upgrade verifications

## Overview

As mentioned in the previous section "Pre-upgrade validations", an ACI fabric upgrade is mainly involving a two-step procedure:

- APIC cluster upgrade.
- ACI nodes upgrade.

This chapter will provide suggestions on what to do and what not to do during the upgrade process.

A few general rules to follow during the upgrade are summarized below:

- Do not force reboot of a device while it is performing an upgrade.
- While performing an upgrade, avoid making any policy change on the ACI fabric.
- Once an upgrade is triggered, do not push further upgrade commands.
- Do not change the fabric topology while the upgrading is ongoing, e.g. do not add or remove nodes or add or remove cables.

Any deviation from the above guidelines may affect the stability of the ACI fabric and the success of the ACI fabric upgrade with irreversible consequences. As explained throughout this chapter, Cisco strongly recommends involving a TAC engineer if the upgrade process fails or results in an unexpected state.

## During the APIC cluster upgrade

During the APIC cluster upgrade process, only one controller will go down for reboot at any given time.

The time of upgrade and reboot of an APIC controller may vary. During the time of reboot, the APIC controller will disappear from the view of other controllers. This is expected. In the screen captures below, APIC3 is being rebooted as result of a triggered controller upgrade.

### APIC firmware during an upgrade - APIC reloading

The screenshot shows the Cisco APIC web interface. The top navigation bar includes the Cisco logo, the text 'APIC', and a user profile 'admin'. Below this is a menu with 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin' (selected), 'Operations', 'Apps', and 'Integrations'. A secondary menu below 'Admin' includes 'AAA', 'Schedulers', 'Historical Record Policies', 'Firmware' (selected), 'External Data Collectors', 'Config Rollbacks', 'Import/Export', and 'Downloads'. The main content area is titled 'Firmware' and has tabs for 'Summary', 'Infrastructure' (selected), 'Images', 'Faults', and 'History'. Under 'Infrastructure', there are sub-tabs for 'Controllers' (selected) and 'Nodes'. The main content displays the upgrade configuration: 'Ignore Compatibility Check: false', 'Target Firmware Version: apic-4.2(1j)', and 'Start time: 2019-10-29 09:57:44.404+00:00'. Below this is a table showing the upgrade progress for two controllers.

ID	Name	Role	Model	Current Firmware	Status	Upgrade Progress
1	bdsol-aci37-apic1	controller	APIC-SERVER-M3	4.2(1j)	Upgraded successfully o...	100%
2	bdsol-aci37-apic2	controller	APIC-SERVER-M3	4.2(1j)	Upgraded successfully o...	100%

## APIC firmware during an upgrade - APIC coming up

The screenshot shows the Cisco APIC web interface. The left sidebar contains navigation options like System, Tenants, Fabric, Virtual Networking, L4-L7 Services, Admin, Operations, Apps, and Integrations. The main content area is titled 'Cluster as Seen by Node' and shows a table of active controllers. The table has the following data:

ID	Name	IP	Admin State	Operational State	Health State	Failover Status	Serial Number	SSL Certificate
1	bdsol-aci37-apid1	10.0.0.1	In Service	Available	Fully Fit	idle	WZP224...	yes
2	bdsol-aci37-apid2	10.0.0.2	In Service	Available	Fully Fit	idle	WZP224...	yes
3	bdsol-aci37-apid3	0.0.0.0	In Service	Unregistered	Not Creat...	idle		yes

If an APIC controller does not come online after 1 hour or more, it is likely that the APIC failed to upgrade.

If this is the case, the following verifications can be made:

- Verify the APIC controller is powered on by checking the device power LED to be on.
- Verify the APIC controller is reachable via ping, SSH, and HTTPS. If this is true, verify the controller accepts user credentials and login is successful.
- Access the KVM console via CIMC in attempt to gather more information about the APIC's current state.

If any of the above checks are failing, open a TAC case so that a Cisco engineer can assist in additional verifications and recovery.

## Post-APIC cluster upgrade

When the APIC Cluster Upgrade is complete, the APIC controllers will be running the target version.

### APIC Firmware after an upgrade

The screenshot shows the APIC Firmware page in the Cisco management interface. The page is titled "Firmware" and has a navigation menu with options: Summary, Infrastructure (selected), Images, Faults, and History. Under the Infrastructure tab, there are sub-tabs for "Controllers" and "Nodes". The "Controllers" sub-tab is active, showing a table of three controllers. Above the table, there are settings: "Ignore Compatibility Check: false", "Target Firmware Version: apic-4.2(1j)", and "Start time: 2019-10-29 09:57:44.404+00:00". The table shows that all three controllers have been upgraded successfully to version 4.2(1j) and are in a "Fully Fit" status.

ID	Name	Role	Model	Current Firmware	Status	Upgrade Progress
1	bdsol-aci37-apic1	controller	APIC-SERVER-M3	4.2(1j)	Upgraded successfully ...	100%
2	bdsol-aci37-apic2	controller	APIC-SERVER-M3	4.2(1j)	Upgraded successfully ...	100%
3	bdsol-aci37-apic3	controller	APIC-SERVER-M3	4.2(1j)	Upgraded successfully ...	100%

The Controllers will be in Active status and the Health Status will be in 'Fully Fit' status.

## APIC Cluster view

The screenshot shows the APIC web interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The left sidebar shows a tree view under 'Controllers' with 'Cluster as Seen by Node' selected. The main content area is titled 'Cluster as Seen by Node' and shows a table of 'Active Controllers'.

ID	Name	IP	Admin State	Operation State	Health State	Failover Status	Serial Number	SSL Certificate
1	bdsol-aci37-apic1	10.0.0.1	In Service	Available	Fully Fit	idle	WZP22...	yes
2	bdsol-aci37-apic2	10.0.0.2	In Service	Available	Fully Fit	idle	WZP22...	yes
3	bdsol-aci37-apic3	10.0.0.3	In Service	Available	Fully Fit	idle	WZP22...	yes

These verifications can also be made via APIC CLI with the commands 'show version' and 'show controller'.

From APIC CLI, controller upgrades can be monitored and verified with the command 'show firmware upgrade status'.

If a controller or switch node is in the process of reloading, it might not appear in the 'show firmware upgrade status' output.

## During the node upgrade

Nodes of the same upgrade group will be upgraded in parallel unless they fall under the following rules:

- Nodes in different Pods can't be upgraded in parallel.
- Nodes in a VPC pair can't be upgraded in parallel.

In those cases, nodes that cannot be upgraded in parallel will be queued for upgrade.

For example, if a VPC pair is part of the same upgrade group, the following will be shown by APIC GUI:

VPC nodes upgrade in same upgrade group

The screenshot shows the APIC GUI interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin' (selected), 'Operations', 'Apps', and 'Integrations'. Below this is a sub-menu with 'AAA', 'Schedulers', 'Historical Record Policies', 'Firmware' (selected), 'External Data Collectors', 'Config Rollbacks', 'Import/Export', and 'Downloads'. The main content area is titled 'Firmware' and has tabs for 'Summary', 'Infrastructure' (selected), 'Images', 'Faults', and 'History'. Under 'Infrastructure', there are sub-tabs for 'Controllers' and 'Nodes' (selected). A toggle for 'Enforce Bootscript Version Validation' is set to off. The main table displays the following data:

ID	Name	Role	Model	Current Firmware	Upgrade Group	Status	Upgrade Progress
Pod1/101	S2P1-Leaf101	leaf	N9K-C93180YC-FX	n9000-14.1(1i)	LeafSwitches	Firmware upgrade queued ...	0%
Pod1/102	S2P1-Leaf102	leaf	N9K-C93180YC-FX	n9000-14.1(1i)	LeafSwitches	Firmware upgrade in progr...	45%
Pod1/201	S2P1-Spine201	spine	N9K-C9332C	n9000-14.1(1i)		Not Scheduled	
Pod1/202	S2P1-Spine202	spine	N9K-C9332C	n9000-14.1(1i)		Not Scheduled	

At the bottom right of the table area, there are 'Reset' and 'Submit' buttons.

## VPC nodes upgrade - Fault record

The screenshot shows the Cisco APIC Admin interface with a modal window displaying a fault record. The fault ID is 8589940098. The description states: "Node: 101, Policy: LeafSwitches, Check constraint: Is VPC peer upgrading?, Result: fail, Details: Rejecting upgrade request from node: 101. VPC peer upgrading, node to retry periodically. Peer node: 10.2.72.64/32". The affected object is "maintupgstatuscont". The fault was created on 2019-10-29T10:36:28.884+00:00 with code E4210684. The cause is "upgrade-scheduler-change" and the change set is "schedulerTick (Old: 25, New: 26)". The action is "state-transition".

In the above case the node 101 is queued for upgrade with the following message:

Node: 101, Policy: pod1\_leaf, Check constraint: Is VPC peer upgrading?, Result: fail, Details: Rejecting upgrade request from node: 101. VPC peer upgrading, node to retry periodically. Peer node: 10.2.72.64/32

The time of upgrade and reboot of a node may vary. During the time of reboot, the node will be shown with an unknown status in the GUI. This is expected. In the screen captures below, node 101 is being rebooted.

## Fabric node upgrade - during upgrade

The screenshot shows the Cisco APIC interface for the 'Firmware' section. The 'Nodes' tab is active, displaying a table of nodes and their upgrade progress. The table has columns for ID, Name, Role, Model, Current Firmware, Upgrade Group, Status, and Upgrade Progress. The 'Upgrade Progress' column for Pod1/101 shows a progress bar at 0%. The 'Status' column for Pod1/101 shows 'Firmware upgrade ...'. The 'Status' column for Pod1/102 shows 'unknown'. The 'Status' column for Pod1/201 and Pod1/202 shows 'Not Scheduled'.

ID	Name	Role	Model	Current Firmware	Upgrade Group	Status	Upgrade Progress
Pod1/101	S2P1-Leaf101	leaf	N9K-C93180YC-FX	n9000-14.1(1i)	LeafSwitches	Firmware upgrade ...	0%
Pod1/102	S2P1-Leaf102	leaf			LeafSwitches	unknown	
Pod1/201	S2P1-Spine201	spine	N9K-C9332C	n9000-14.1(1i)		Not Scheduled	
Pod1/202	S2P1-Spine202	spine	N9K-C9332C	n9000-14.1(1i)		Not Scheduled	

If a node does not come online after 1 hour or more, it is likely that the node failed to upgrade.

If this is the case, the following verifications can be made:

- Verify the node is powered on by checking the switch chassis LED to be on.
- Verify the node is reachable via ping and SSH. If this is true, verify the node accepts user credentials and login is successful.
- Access the node via console port in attempt to glean additional information about the current state.

If any of the above checks are failing, open a TAC case so that a Cisco engineer can assist in additional verifications and recovery.

## Post-node upgrade

When a node upgrade is performed, the node will be running the target version.

### Node firmware - post upgrade

The screenshot shows the APIC interface for the 'Firmware' section. The 'Infrastructure' tab is selected, and the 'Nodes' sub-tab is active. A table lists four nodes, all of which have been upgraded successfully to the target firmware version n9000-14.2(2e). The upgrade progress for each node is shown as a green bar at 100%.

ID	Name	Role	Model	Current Firmware	Upgrade Group	Status	Upgrade Progress
Pod1/101	S2P1-Leaf101	leaf	N9K-C93180YC-FX	n9000-14.2(2e)	LeafSwitches Target FW: n9000-14.2(2e)	Upgraded successfully on 2019...	100%
Pod1/102	S2P1-Leaf102	leaf	N9K-C93180YC-FX	n9000-14.2(2e)	LeafSwitches Target FW: n9000-14.2(2e)	Upgraded successfully on 2019...	100%
Pod1/201	S2P1-Spine201	spine	N9K-C9332C	n9000-14.2(2e)	SpineSwitches Target FW: n9000-14.2(2e)	Upgraded successfully on 2019...	100%
Pod1/202	S2P1-Spine202	spine	N9K-C9332C	n9000-14.2(2e)	SpineSwitches Target FW: n9000-14.2(2e)	Upgraded successfully on 2019...	100%

This verification can also be made via CLI with the command 'show version'.

From APIC CLI, node upgrades can be monitored and verified with the command 'show firmware upgrade status'.

If a controller or switch node is in the process of reloading, it might not appear in the 'show firmware upgrade status' output.

## FPGA / EPLD / BIOS

There are various **methods** to load a firmware image on a leaf or a spine in order to upgrade it.

- 1 Perform a policy upgrade/downgrade through the APIC.
- 2 Perform an NXOS to ACI conversion.
- 3 Load an image while at the loader prompt through USB or TFTP.
- 4 Transfer an image (SCP, SFTP, USB etc.) to '/bootflash' of the switch and use the 'setup-bootvars.sh <image>' command followed by reload of the device.

**The first method** will always assure that the FPGA / EPLD and BIOS images are correctly upgraded and this is always the supported way to perform an upgrade / downgrade.

It is strongly advised to use a policy upgrade / downgrade via the APIC GUI. Not doing so could lead to FPGA, EPLD, and BIOS related faults and in the worst case scenario, result in devices that cannot activate their front-panel ports. In such a scenario the device would be unable to join the fabric, preventing the administrator from being able to perform corrective actions via the APIC GUI.

Should method 2, 3, or 4 be used, there is a risk of receiving the following error: 'F1582 FPGA version mismatch detected. Running version:<0x(z)> Expected version:<0x(y)>'. The way to resolve this issue is explained in the section "Device replacement" in the chapter "Fabric discovery".

# CIMC

## Overview

This section will cover the **Cisco Integrated Management Controller (CIMC)** configuration, hardware items it is capable of monitoring, and upgrade procedure. The CIMC is a component of all UCS C-series used as a lights-out management interface with the intention of providing detailed information about the hardware components in the chassis. It provides capabilities such as changing the power state of the server, remote console connection, SNMP alerts for hardware monitoring etc. The CIMC should be upgraded at regular intervals to ensure compatibility of hardware firmware with the APIC Operating System (OS).

### Hardware models

As of the writing of this book, there are 3 generations of APIC supported with ACI.

#### APIC Generations

Appliance Generation	Part Number	UCS Model Base
<b>Third</b>	APIC-L3	UCSC-C220-M5
	APIC-M3	UCSC-C220-M5
<b>Second</b>	APIC-L2	UCSC-C220-M4
	APIC-M2	UCSC-C220-M4
<b>First</b>	APIC-L1	UCSC-C220-M3
	APIC-M1	UCSC-C220-M3

## CIMC configuration

The CIMC server plays a critical role when troubleshooting the APIC server if the APIC is found to be unreachable through its management address. In order to use CIMC under such conditions, it must be configured for remote access (IP address, password, etc.). Note that this is typically configured using a physical keyboard and monitor connected to the server.

The CIMC can be configured upon boot up of the APIC appliance. This is done by pressing the **F8** key at the **BIOS POST** screen.

[APIC boot up screen](#)



While in the CIMC configuration page, parameters to allow remote access can be configured according to the requirements of the OOB network. **Dedicated mode** is

required for correct operation in the ACI fabric.

### CIMC Configuration Utility

```

CIMC Configuration Utility  Version 1.7  Cisco Systems, Inc.
*****
NIC Properties
NIC mode                               NIC redundancy
Dedicated:      [X]                   None:           [X]
Shared LOM:     [ ]                   Active-standby:[ ]
Cisco Card:     [ ]                   Active-active:  [ ]
Shared LOM Ext: [ ]

IPV4 (Basic)                            Factory Defaults
DHCP enabled:   [ ]                   CIMC Factory Default:[ ]
CIMC IP:        10.48.22.74           Default User (Basic)
Subnetmask:     255.255.255.0         Default password:
Gateway:        10.48.22.100         Reenter password:

VLAN (Advanced)                          Port Profile
VLAN enabled:   [ ]                   Reset:          [ ]
VLAN ID:        1                     Name:
Priority:        0

Port Properties
Auto Negotiation:  [X]
Speed[1000/100 Mbps]:  100
Duplex mode[half/full]: full
*****
<Up/Down>Selection  <F10>Save  <Space>Enable/Disable  <F5>Refresh  <ESC>Exit

```

Once the configuration is complete, the CIMC should be accessible via HTTPS and SSH.

Within the CIMC, several hardware items can be viewed and managed to ensure proper operation of the APIC. The following discusses items specific to an APIC appliance.

## Serial over LAN (SoL)

Serial over LAN is a feature of CIMC to allow console redirection to the user's SSH session. This allows the user to view a remote console of the APIC like the vKVM. This can be used as an alternative if the user has issues working with the **Java** or **HTML5 vKVM** console.

To enable the SoL feature, use the following commands on the CIMC CLI.

```
C220-FCH1930V2Z7# scope sol
C220-FCH1930V2Z7 /sol # set enabled yes
C220-FCH1930V2Z7 /sol # commit
C220-FCH1930V2Z7 /sol # show
Enabled Baud Rate(bps) Com Port SOL SSH Port
-----
yes      115200      com0      2400
```

Once enabled, use the following command to connect to the Serial console.

```
C220-FCH1930V2Z7# connect host
CISCO Serial Over LAN:
Press Ctrl+x to Exit the session

apic1 login:
```

Please do know enabling SOL will disable the onboard hardware console port.

## LLDP

LLDP is an essential protocol to fabric discovery and connectivity between the APIC and leaf. The **Cisco VIC adapter** used for the fabric ports of the APIC are also capable generating LLDP packets. APIC appliances ship with LLDP disabled on the VIC by default and will not operate correctly if it is enabled. This is because when LLDP is enabled on the VIC, LLDP packets will be sent and received by the VIC itself rather than passing the discovery packets up to the APIC OS. In such a scenario, the APIC would fail to discover the leaf switches to which it is connected. The LLDP setting on the VIC can be found and verified via SSH with the following commands.

```
C220-FCH1930V2Z7# scope chassis
C220-FCH1930V2Z7 /chassis # scope adapter 1
C220-FCH1930V2Z7 /chassis/adapter # show detail
PCI Slot 1:
  Product Name: UCS VIC 1225
  Serial Number: FCH19277ZTT
  Product ID: UCSC-PCIE-CSC-02
  Adapter Hardware Revision: 6
  Current FW Version: 4.1(3a)
```

```

VNTAG: Disabled
  FIP: Disabled
  LLDP: Disabled
  Configuration Pending: no
  Cisco IMC Management Enabled: no
  VID: V03
  Vendor: Cisco Systems Inc
  Description:
  Bootloader Version: 4.0(1e)
  FW Image 1 Version: 4.1(3a)
  FW Image 1 State: RUNNING ACTIVATED
  FW Image 2 Version: 4.0(1e)
  FW Image 2 State: BACKUP INACTIVATED
  FW Update Status: Idle
  FW Update Error: No error
  FW Update Stage: No operation (0%)
  FW Update Overall Progress: 0%

```

If this is for some reason set to Enabled, it can be updated with the following commands.

```

C220-FCH1930V2Z7 /chassis/adapter # set lldp disabled
C220-FCH1930V2Z7 /chassis/adapter *# commit
New VNIC adapter settings will take effect upon the next server reset

```

Modification of LLDP setting on the VIC will only take effect after a reboot of the APIC.

## RAID/HDD/SSD

The APIC appliance comes pre-configured with RAID across its multiple disks. The health status of the virtual drive groups can be seen in the CIMC GUI under the Storage tab. The RAID configuration should be as follows.

APIC RAID layout

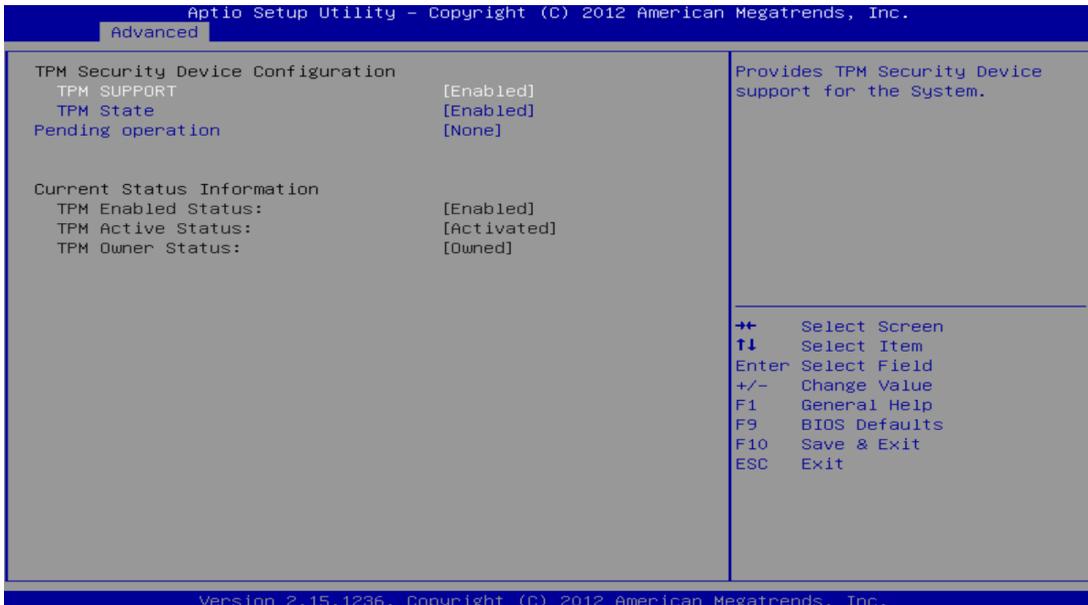
RAID Level	Drives	Media Type
1	2 and 3	HDD
0	1	SSD

## TPM

The **Trusted Platform Module (TPM)** is a hardware component used to verify and authenticate the server. For the APIC to boot and operate as expected, the TPM must be enabled, activated, and owned from the BIOS perspective. If the TPM is not in this state, the APIC may fail to boot or upgrade.

To verify the TPM state, check under **Advanced > Trusted Computing** in the BIOS Setup.

### BIOS Advanced tab view



The TPM settings should not be modified under any circumstances without Cisco TAC involvement as it may render the appliance unusable.

## Host Upgrade Utility (HUU)

The **Cisco Host Upgrade Utility** (HUU) is a tool that can be used to upgrade the CIMC firmware on the APIC appliance as well as other UCS C-series chassis components. The tool will update components such as VIC adapter, RAID controller, and BIOS version of the APIC. It is recommended to upgrade all components available when performing this task.

To get started with upgrading the CIMC firmware via HUU, identify the proper UCS model base for APIC from the chart in the **Hardware Models** section of this chapter. Once identified, download the recommended firmware based on the ACI version. The full list of CIMC firmware supported, and recommended version to be used, is available in the **Release Notes** of the APIC software.

Before beginning the CIMC upgrade, ensure the APIC cluster is **fully fit**. The upgrade should only be performed on 1 APIC at a time and the APIC to undergo CIMC upgrade should be **decommissioned** from the cluster while the maintenance is performed.

To load the image onto the CIMC, two options are available. One is via a HTTP/HTTPS mount and one is via vKVM mount. The basic steps of both procedures will be outlined below.

### vMedia Mount

- 1 Log into CIMC CLI via SSH.
- 2 Enable the vMedia feature if not already done.

```
C220-FCH1930V2Z7# scope vmedia
C220-FCH1930V2Z7 /vmedia # set enabled yes
C220-FCH1930V2Z7 /vmedia *# commit
C220-FCH1930V2Z7 /vmedia # show
Encryption Enabled Enabled Active Sessions Low Power USB Enabled
-----
no                yes          1                yes
```

- 3 Map the HUU via the HTTP/HTTPS path. For ease of use, the HUU can be copied to the '/data/techsupport' directory of another APIC and will automatically be hosted via HTTPS. The example below will show the HUU mounted on APIC2 of the cluster but the number in the URL will change depending on the APIC hosting the file.

```
C220-FCH1930V2Z7 /vmedia # map-www mnt https://10.122.141.117/files/2/techsupport/ ucs-c220-huu-3.0.4j.iso
Server username: admin
Server password:
Confirm password:
C220-FCH1930V2Z7 /vmedia # show mappings detail
Volume mnt:
  Map-Status: OK
  Drive-Type: CD
  Remote-Share: https://10.122.141.117/files/2/techsupport/
  Remote-File: ucs-c220-huu-3.0.4j.iso
  Mount-Type: www
  Mount-Options: "noauto,username=admin,password=*****"
```

- 4 Reboot the APIC via the **acdiag reboot** command.
- 5 Upon BIOS POST screen, use the **F6** key to enter the **Boot Selection** menu. If prompted for a password, the default is 'password'.
- 6 Select the **Cisco CIMC-Mapped vDVD** to boot the HUU.
- 7 The HUU will begin to inventory the chassis and after several minutes, the HUU User Interface will appear.
- 8 Select the '**Upgrade All**' option to begin the upgrade of CIMC firmware.
- 9 When completed, the user can quit the HUU and reboot using the '**Exit**' option. If a BIOS upgrade is done, the APIC will automatically reboot to complete this process.

## vKVM Mount

- 1 Log into CIMC HTTPS GUI.
- 2 Launch the vKVM console using HTML5 or Java.

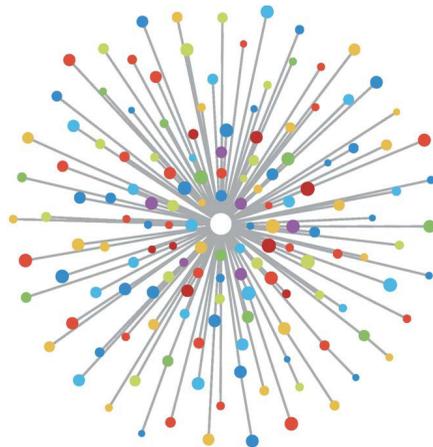
- 3 Mount the selected HUU via **Virtual Media** on the **vKVM console**.
- 4 Reboot the APIC via the '**acidiag reboot**' command.
- 5 Upon BIOS POST screen, use the **F6** key to enter the **Boot Selection** menu. If prompted for a password, the default is 'password'.
- 6 Select the **Cisco vKVM-Mapped vDVD** to boot the HUU.
- 7 The HUU will begin to inventory the chassis and after several minutes, the HUU User Interface will appear.
- 8 Select the '**Upgrade All**' option to begin the upgrade of APIC firmware.
- 9 When completed, the user can quit the HUU and reboot using the '**Exit**' option. If a BIOS upgrade is done, the APIC will automatically reboot to complete this process. The vKVM will disconnect as well.

After the HUU update has completed and the APIC is booted again, the user can **recommission** it from another APIC in the cluster. Additionally, some hardware components may reset to their default values (such as LLDP being activated on the VIC). If the APIC fails to boot or not able to rejoin the cluster, validate the settings called out for components listed previously in this section.





# Acronyms



# Acronyms

ACI - Application Centric Infrastructure  
ACL - Access Control List  
AEP - Access Entity Profile  
APIC - Application Policy Infrastructure Controller  
AP - Application Profile  
ARP - Address Resolution Protocol  
AS - Autonomous System  
ASIC - Application-Specific Integrated Circuit  
ASN - Autonomous System Number  
AVS - Application Virtual Switch  
BD - Bridge Domain  
BGP - Border Gateway Protocol  
BIOS - Basic Input/Output System  
BL - Border Leaf  
BOOTP - Bootstrap Protocol  
BUM - Broadcast, Unknown-Unicast, and Multicast  
CAM - Content Addressable Memory  
CDP - Cisco Discovery Protocol  
CIDR - Classless Inter-Domain Routing  
CIMC - Cisco Integrated Management Controller  
CLI - Command Line Interface  
COOP - Council Of Oracle Protocol  
CPLD - Complex Programmable Logic Devices  
CRC - Cyclic Redundancy Check  
DHCP - Dynamic Host Configuration Protocol  
DNS - Domain Name System  
DR - Designated Router  
DSCP - Differentiated Services Code Point  
DVS - Distributed Virtual Switch  
ECMP - Equal Cost Multipath Routing  
EIGRP - Enhanced Interior Gateway Routing Protocol  
ELAM - Embedded Logic Analyzer Module

EP - EndPoint  
EPG - EndPoint Group  
EPLD - Electronic Programmable Logic Device  
ERSPAN - Encapsulated Remote Switched Port Analyzer  
ETEP - External Tunnel Endpoint  
EVPN - Ethernet Virtual Private Network  
FC - Fibre Channel  
FCS - Frame Check Sequence  
FHRP - First Host Redundancy Protocol  
FPGA - Field Programmable Gate Arrays  
FQDN - Fully Qualified Domain Name  
FTAG - Forwarding Tag  
GIPo - Group Internet Protocol Outer  
GRE - Generic Routing Encapsulation  
HTTP - HyperText Transfer Protocol  
HTTPS - HyperText Transfer Protocol Secure  
HUU - Host Update Utility  
ICMP - Internet Control Message Protocol  
IGMP - Internet Group Management Protocol  
IGP - Interior Gateway Protocol  
IP - Internet Protocol  
IPN - Inter-Pod Network  
IS-IS - Intermediate System to Intermediate System  
ISN - Inter-Site Network  
KVM - Keyboard, Video, and Mouse  
LACP - Link Aggregation Control Protocol  
LAN - Local Area Network  
LBT - Load-Based Teaming  
LC - Line Card  
LED - Light Emitting Diode  
LLDP - Link Layer Discovery Protocol  
LSA - Link State Advertisement  
LSDB - Link State Database  
MAC address - Media Access Control address  
MP-BGP - Multiprotocol Border Gateway Protocol

MSDP - Multicast Source Discovery Protocol  
MTU - Maximum Transmission Unit  
NAT - Network Address Translation  
NIC - Network Interface Card  
NSSA - Not-So-Stubby Area  
NTP - Network Time Protocol  
NXOS - Nexus Operating System  
OID - Object Identifier  
OIF - Outgoing Interface  
OOB - Out-of-Band  
OSPF - Open Shortest Path First  
PBR - Policy Based Redirect  
PCI - Peripheral Component Interconnect  
PCIE - Peripheral Component Interconnect Express  
PC - Port Channel  
PID - Product Identifier  
PIM BiDir - Protocol Independent Multicast Bi-Directional  
POAP - Power On Auto Provisioning  
PSU - Power Supply Unit  
PTEP - Physical Tunnel Endpoint  
RAID - Redundant Array of Independent Disks  
RBAC - Role-Based Access Control  
RIB - Routing Information Base  
RMA - Return Merchandise Authorization  
RPF - Reverse Path Forwarding  
RP - Rendezvous Point  
RR - Route Reflector  
SCP - Secure Copy Protocol  
SDK - Software Development Kit  
SDN - Software-Defined Networking  
SFP - Small Form-Factor Pluggable  
SFTP - Secure File Transfer Protocol  
SNMP - Simple Network Management Protocol  
SPAN - Switched Port Analyzer  
SSD - Solid State Drive

SSL - Secure Sockets Layer  
STP - Spanning-Tree Protocol  
SUP - Supervisor  
SVI - Switched Virtual Interface  
TAC - Technical Assistance Center  
TCAM - Ternary Content-Addressable Memory  
TCP - Transmission Control Protocol  
TEP - Tunnel End Point  
TFTP - Trivial File Transfer Protocol  
TLV - Type Length Value  
TOS - Type Of Service  
TPM - Trusted Platform Module  
TTL - Time To Live  
UDP - User Datagram Protocol  
URL - Uniform Resource Locator  
UTC - Universal Time Coordinated  
UUID - Universally Unique Identifier  
VIC - Virtual Interface Card  
VIP - Virtual Internet Protocol  
VLAN - Virtual Local Area Network  
VM - Virtual Machine  
VMK - Virtual Machine Kernel  
VMM - Virtual Machine Manager  
VNI - Virtual Network Instance  
VNIC - Virtual Network Interface Card  
VNID - Virtual Network Identifier  
VNTAG - Virtual Network Tag  
VPC - Virtual Port Channel  
VPNv4/v6 - Virtual Private Network version 4/6  
VRF - Virtual Routing and Forwarding  
VTEP - Virtual Tunnel End Point  
VXLAN - Virtual Extensible Local Area Network

