Configuring ACI WAN Interconnect

This chapter contains the following sections:

VXLAN EVPN - MPLS L3VPN for ACI Fabric

Overview of VXLAN EVPN - MPLS L3VPN for ACI Fabric

ACI WAN Interconnect is a multi-platform, multi-OS Data Center Interconnect (DCI) architecture. It connects multi-tenant VXLAN data center fabrics over L3VPN.

The Cisco Application Centric Infrastructure (ACI) allows application requirements to define the network. This architecture simplifies, optimizes, and accelerates the entire application deployment life cycle.

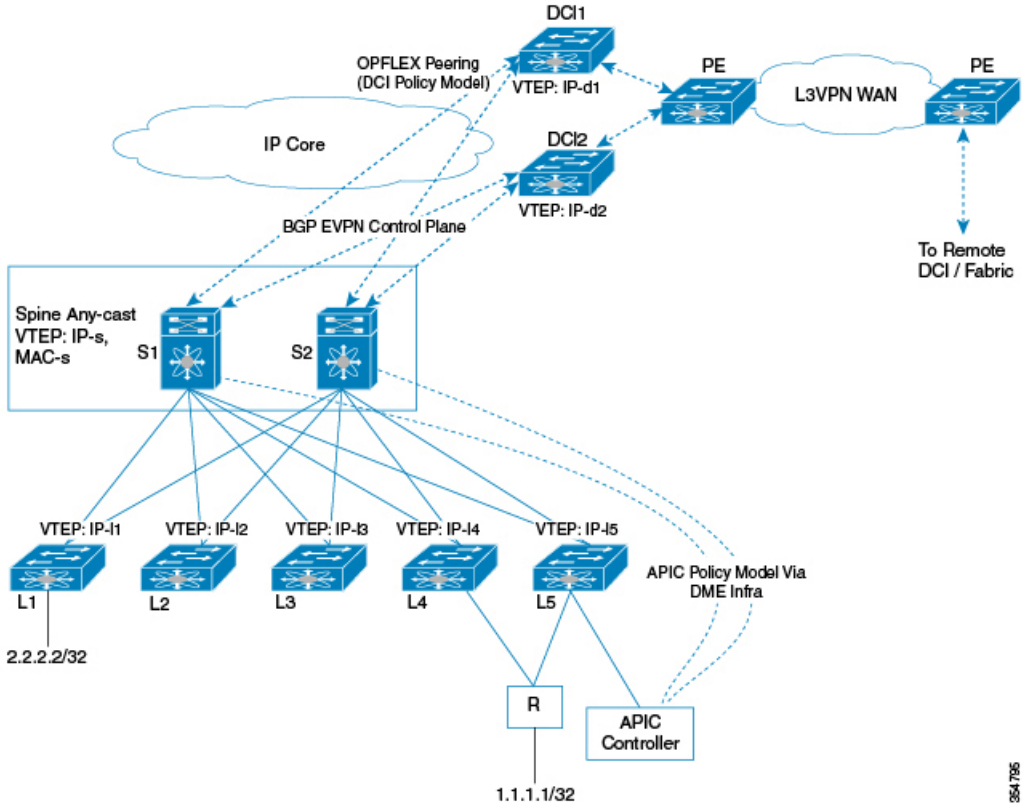
The ACI fabric includes switches with the APIC to run in the leaf/spine ACI fabric mode. These switches form a “fat-tree” network by connecting each leaf node to each spine node; all other devices connect to the leaf nodes. The APIC manages the ACI fabric. The recommended minimum configuration for the APIC is a cluster of three replicated hosts. The APIC fabric management functions do not operate in the data path of the fabric.

- For more details on ACI, refer to Cisco Application Centric Infrastructure Fundamentals guide and Cisco ACI Basic Configuration Guide.

The VXLAN EVPN data center fabric can be connected across Layer 3 boundaries (to external sites and back) using MPLS L3VPN, VRF IP Routing (VRF Lite), or LISP as the mechanism of transport outside the VXLAN fabric.

The ACI WAN Interconnect feature requires the MPLS package and the LAN enterprise package licenses.

Figure 1: ACI Fabric & L3VPN Hand-off



The MPLS L3VPN hand off scenario is explained below.

- •BGP-EVPN peering from DCI gateways to two spines in a POD.
- Spines advertise host OR prefix routes for hosts directly behind a leaf, with the Spine Any-cast IP VTEP (IP-s) as the next-hop. These are mostly public BD subnets advertised to outside world.
- Spines can relay a transit route advertised by ACI leafs with leaf VTEPs as next-hops to be used as ECMP paths.
- North-to-South traffic tunneled from DCI to Spine any-cast IP (can land on any of the Spines) or the ECMP is tunneled directly to advertise ACI-leaf VTEPs.
- North-to-South traffic tunneled to Spine will get routed on spine-to-leaf based on /32 lookup.
- Routes advertised from DCI to Spine will get reflected to leaves with the DCI VTEP as the next-hop.
- South-to-North traffic will get routed on the leaves, and ECMP is tunneled directly to the two DCIs.
- Downstream assigned per-VRF VNIDs are advertised by DCI and ACI VTEPs.
- DCI tenant configuration object model are pushed from APIC to Spine to DCI via the OpFlex framework.
- Spines advertise public BD subnet host or prefix routes for hosts directly behind a leaf, with the Spine Any-cast IP VTEP (IP-s) as the next-hop.

- Physical and underlay L3 connectivity between the DCIs and Spines can be via an infrastructure IP network in between or via direct layer 3 sub-interfaces.

Spine – DCI Connectivity

ACI Spines is directly connected or connected via an inter-POD network router to DCI gateways. Underlay connectivity being direct or via an intermediate router does not have any bearing on the DCI gateway functions.

Spine – DCI BGP EVPN Session

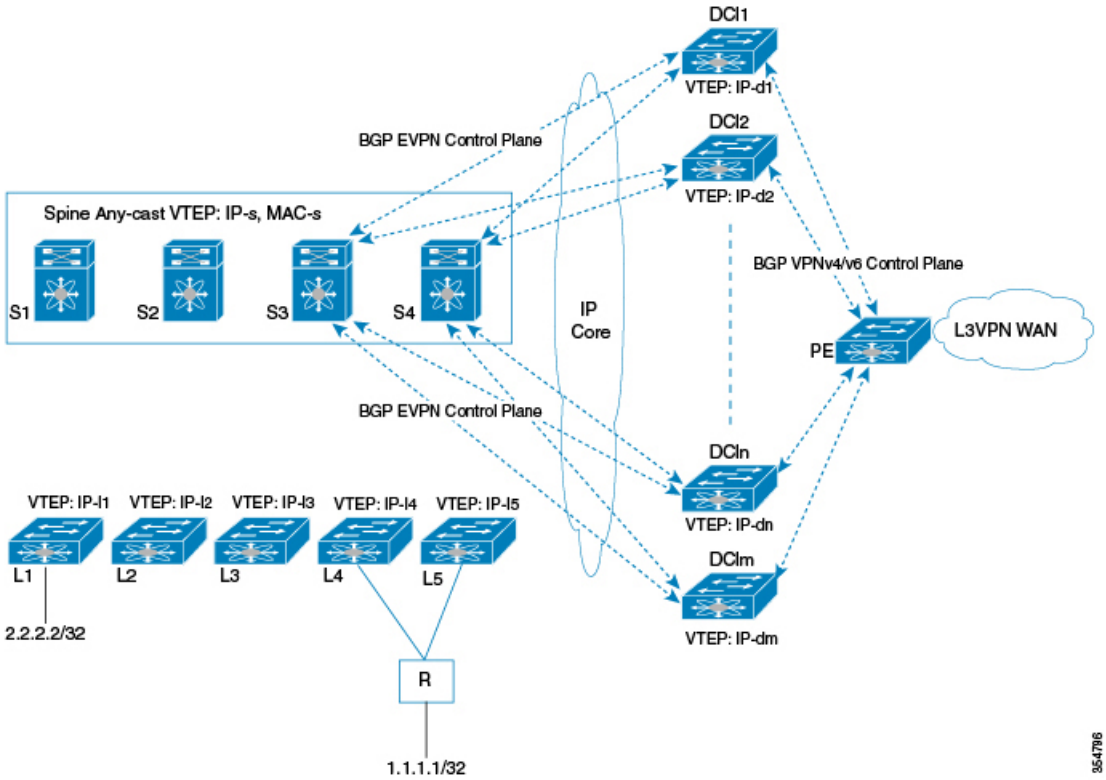
BGP session between the Spine and DCI gateway can be eBGP or iBGP. eBGP is the commonly used topology.

The MPLS-L3VPN hand-off for ACI fabric can be deployed using one of the following topologies:

- Single POD with multiple DCI gateways
- Multi-POD with shared DCI gateway
- Multi-POD with Separate DCI gateway

Single POD With Multiple DCI Gateways

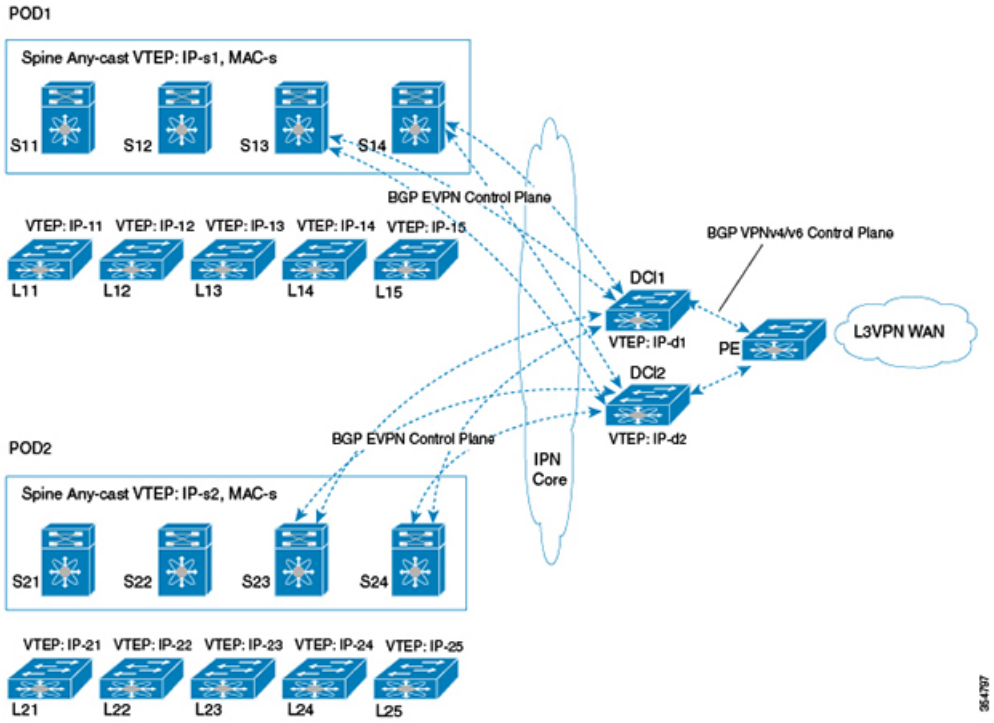
Figure 2: Single POD with Multiple DCI Gateways



This topology is used if VRF scale within the single POD is more that the VRF scale supported on a single DCI gateway. A set of VRFs are imported and advertised on one DCI pair, while another set of VRFs could be present on another DCI pair. Fabric spines advertise all routes to all DCI pairs, but only configured VRF routes are imported and advertised towards L3VPN PE on the respective DCIs.

Multi-POD With Shared DCI Gateways

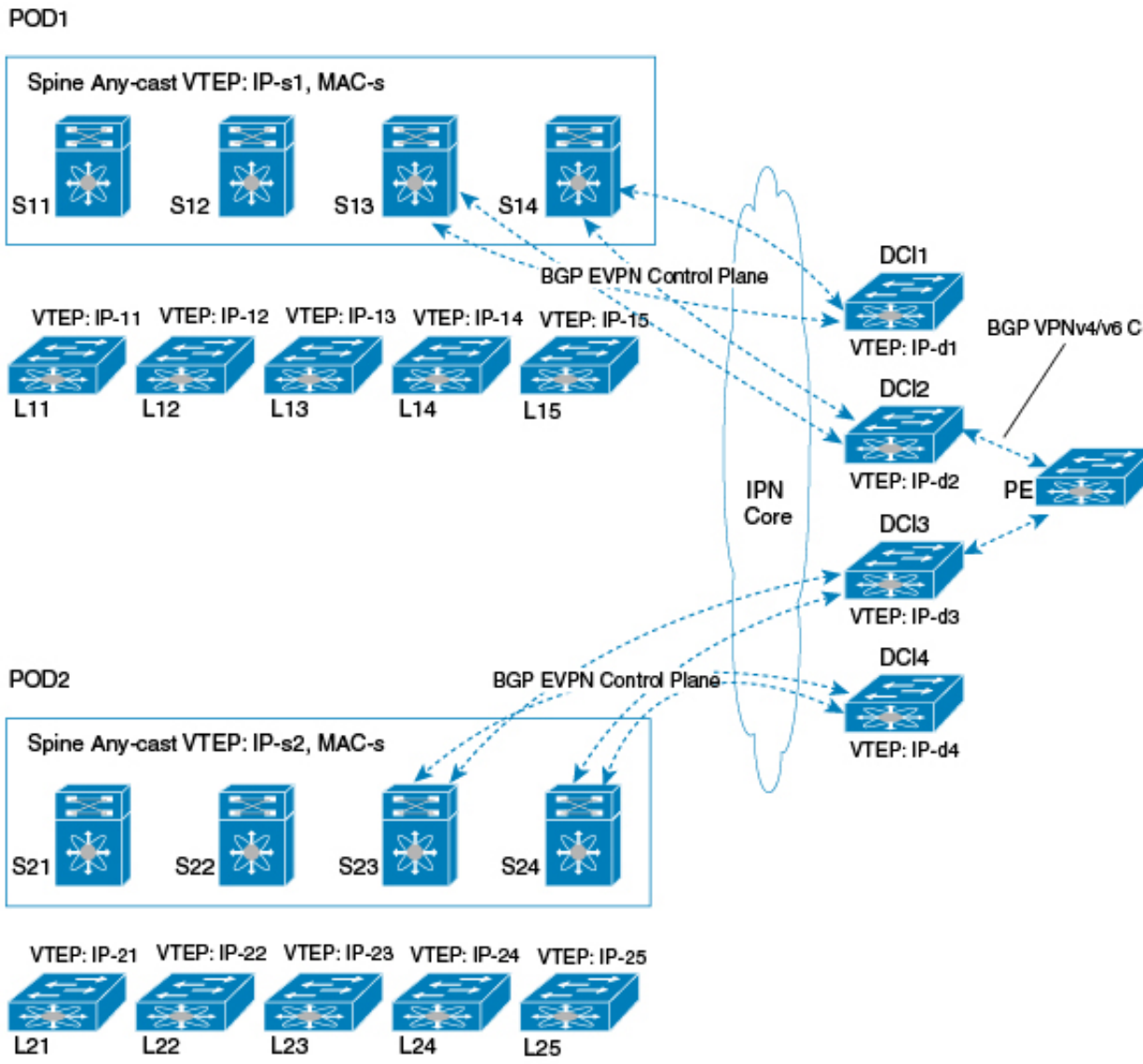
Figure 3: Multi-POD with Shared DCI Gateway



In this topology multiple PODs share the same DCI gateway. The DCI pair imports and advertises VRF routes from multiple POD spines. DCI pair has underlay connectivity to multiple PODs over an inter-POD network underlay.

Multi-POD With Separate DCI Gateways

Figure 4: Multi-POD with Separate DCI Gateways



This is a regular multi-POD topology where separate PODs use dedicated DCI gateways.

Configuration for VXLAN EVPN - MPLS L3VPN for ACI Fabric using OpFlex is described in the following section.

OpFlex DCI Auto-Configuration

Cisco OpFlex is a southbound protocol in a software-defined network (SDN) designed to facilitate the communications between the SDN Controller and the infrastructure (switches and routers). The goal is to

create a standard that enables policies to be applied across physical and virtual switches/routers in a multi-user environment.

For more details on OpFlex refer to OpFlex: An Open Policy Protocol White Paper.

To enable automation of fabric facing tenant configuration on the DCI, DCI interfaces with the fabric as an external Policy Element (PE) that talks to ACI fabric spine acting as a proxy-Policy Repository (PR) for DCI specific policy information. An OpFlex policy framework is used between the spines and the DCIs to distribute this DCI policy model from the fabric to the DCI gateways. DCI uses this policy information pushed from the spine to auto generate fabric facing per-tenant configuration.

ACI spine in turn derives this DCI object model from the concrete object model (object store) populated on the Spine through the ACI DME infrastructure. APIC controller (via DME infra) pushes a logical model that results in a resolved concrete model on the Spine Policy Element. Spine gleans specific attributes required to instantiate the ACI WAN Interconnect DCI service for individual tenants from this resolved concrete model and populates a per-DCI object-model that is distributed to individual DCIs via the OpFlex framework. Spine essentially acts a proxy on behalf of the fabric to push per-tenant DCI policies to the DCI that acts as an external policy element to the fabric.

Note that this PR – PE contract between the fabric and the DCI is limited to fabric facing per-tenant provisioning and not a contract for management functions in general. All remaining configuration, including WAN facing configuration, as well as all other management, operational aspects on the DCI will continue to work independent of this PR – PE contract with the fabric, via existing mechanisms.

Interconnect Policy Provisioning (IPP)

Interconnect Policy Provisioning (IPP) enables automation of fabric facing per-tenant provisioning on the DCI gateway.

The IPP utilizes OpFlex to push policies from ACI fabric to the DCI gateway. Using these policy attributes, HMM auto-config is triggered to apply the profile along with the attributes to provision the required fabric-facing configurations.

OpFlex Peering and Multi-POD

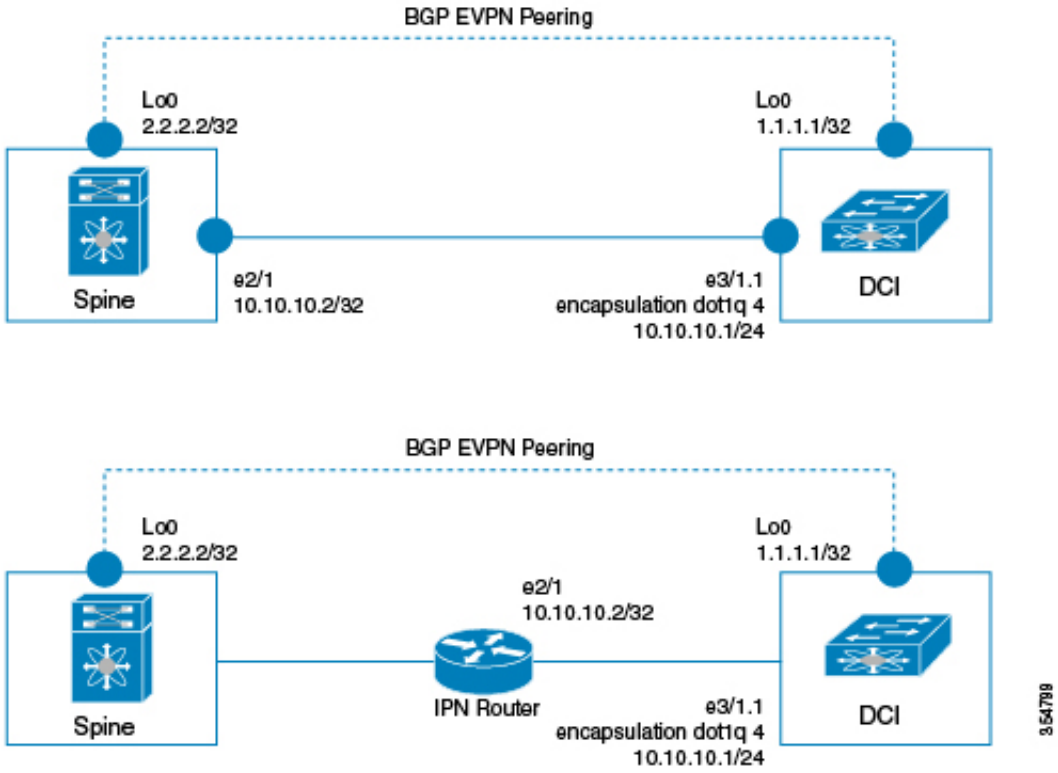
In a multi-POD topology, where the same DCI pair peers with multiple PODs, each POD would be configured as a separate fabric ID and would result in a separate OpFlex framework to be instantiated with the respective spine peers in that fabric that would result in OpFlex sessions to the spine peers within each POD.

Each OpFlex framework translates to a separate managed object database that is populated as a result of policy information distributed from respective spines.

In a scenario where the same L3 domain is spread across the two PODs, DCI can receive updates for the same VRF from multiple OpFlex frameworks, possibly with different RTs. DCI handles this multiple update scenario by appending the route targets for the POD if the fabric facing local VRF configuration has already been instantiated.

DCI Auto-Configuration Scenario

Figure 5: DCI Auto-Configuration



DCI could have the following scenarios:

- Underlay L3 connectivity to the spine via an Inter-POD-Network router in a multi-POD topology
- Directly connected to the spine

Spine — IPN Router — DCI (multi-pod topology)

DCI would be part of an external subnet that is reachable via an inter-pod network/router and not part of the infra subnet (that is administered via APIC). External subnets are administered outside of APIC and would have the IP addresses administered manually.

Spine — DCI (DCI directly connected to multiple spines)

A separate subnet is allocated for all DCIs to be on, and this subnet is not administered via APIC/DHCP. DCI and spine interface IP addresses on this subnet are administered manually.

Essentially, in both topologies, DCI underlay interface that carries OpFlex/ BGP control plane/VXLAN data plane traffic is assumed to be not on the infra subnet that requires IP address to be obtained via DHCP. L3 reachability to ACI VTEPs, BGP peers, and OpFlex proxy on the ACI infra subnet is through the underlay routing to/from this external DCI subnet.

The following sections describe the OpFlex configuration steps.

1) One time configuration (as described below) for the following is done manually:

- DCI underlay connectivity
- Routing
- BGP-EVPN peering to the spines
- BGP-IPVPN peering to the WAN PE

```
# enable features
install feature-set mpls
install feature-set fabric
feature-set mpls
feature-set fabric
feature fabric forwarding
fabric forwarding switch-role dci-node border
nv overlay evpn
feature bgp
feature interface-vlan
feature nv overlay
feature vni
feature ospf
feature ipp
feature mpls l3vpn
feature mpls ldp


#BGP Fabric and WAN Peering
router bgp 65000
  address-family l2vpn evpn
    allow-vni-in-ethertag
  ! EVPN Neighbor
  neighbor 2.2.2.1 remote-as 75000
    address-family l2vpn evpn
      import vpn unicast reoriginate
  ! WAN Peering
  neighbor 11.11.11.1 remote-as 65000
    update-source loopback 0
    address-family vpnv4 unicast
      import l2vpn evpn reoriginate
```

Note The **allow-vni-in-ethertag** configuration allows EVPN Route Type 2 routes to be received from the ACI spine devices.

```
# DCI TEP IP
interface loopback0
  ipv4 address 1.1.1.1/32

# underlay fabric facing L3 interface
interface e3/1
  ipv4 address 10.10.10.1/24

# VXLAN local TEP
interface NVE 1
  source-interface loopback0
  host-reachability protocol bgp
  unknown-peer-forwarding enable
  vxlan udp port 48879

# underlay routing
router ospf
  area 0
    interface loopback0
    interface e3/1
```

```
# DCIs learn reachability to all ACI TEP IPs via OSPF or ISIS
router ospf 100
  router-id 40.0.0.9
  area 0.0.0.100 nssa
# Configuring a peer in downstream (vni) mode
interface nve 1
[no] vni assignment downstream [all]
        Peer-ip <ip address 1>
        Peer-ip <ip address 2>
     ……….
     Peer-ip <ip address 3>

# Changing the default forwarding behavior

interface nve <nve-int-number>
 [no] unknown-peer-forwarding enable
```

2) Configuring profile templates

The following config-profile templates are manually configured so that IPP can leverage HMM auto-config functionality to instantiate the profiles for the VRF tenant.

```
# MPLS L3VPN hand-off common profile
configure profile vrf-common-mpls-l3vpn-dc-edge
  vrf context $vrfName
    vni $include_vrfSegmentId
    rd auto
    address-family ipv4 unicast
      route-target import $include_client_import_ipv4_bgpRT_1 evpn
      route-target export $include_client_export_ipv4_bgpRT_1 evpn
      route-target import $include_client_import_ipv4_bgpRT_2 evpn
      route-target export $include_client_export_ipv4_bgpRT_2 evpn
      route-target import $include_client_import_ipv4_bgpRT_3 evpn
      route-target export $include_client_export_ipv4_bgpRT_3 evpn
      route-target import $include_client_import_ipv4_bgpRT_4 evpn
      route-target export $include_client_export_ipv4_bgpRT_4 evpn
      route-target import $include_client_import_ipv4_bgpRT_5 evpn
      route-target export $include_client_export_ipv4_bgpRT_5 evpn
      route-target import $include_client_import_ipv4_bgpRT_6 evpn
      route-target export $include_client_export_ipv4_bgpRT_6 evpn
      route-target import $include_client_import_ipv4_bgpRT_7 evpn
      route-target export $include_client_export_ipv4_bgpRT_7 evpn
      route-target import $include_client_import_ipv4_bgpRT_8 evpn
      route-target export $include_client_export_ipv4_bgpRT_8 evpn
    address-family ipv6 unicast
      route-target import $include_client_import_ipv6_bgpRT_1 evpn
      route-target export $include_client_export_ipv6_bgpRT_1 evpn
      route-target import $include_client_import_ipv6_bgpRT_2 evpn
      route-target export $include_client_export_ipv6_bgpRT_2 evpn
      route-target import $include_client_import_ipv6_bgpRT_3 evpn
      route-target export $include_client_export_ipv6_bgpRT_3 evpn
      route-target import $include_client_import_ipv6_bgpRT_4 evpn
      route-target export $include_client_export_ipv6_bgpRT_4 evpn
      route-target import $include_client_import_ipv6_bgpRT_5 evpn
      route-target export $include_client_export_ipv6_bgpRT_5 evpn
      route-target import $include_client_import_ipv6_bgpRT_6 evpn
      route-target export $include_client_export_ipv6_bgpRT_6 evpn
      route-target import $include_client_import_ipv6_bgpRT_7 evpn
      route-target export $include_client_export_ipv6_bgpRT_7 evpn
      route-target import $include_client_import_ipv6_bgpRT_8 evpn
      route-target export $include_client_export_ipv6_bgpRT_8 evpn*
  router bgp $asn
    vrf $vrfName
      address-family ipv4 unicast
        advertise l2vpn evpn
        label-allocation-mode per-vrf
      address-family ipv6 unicast
        advertise l2vpn evpn
        label-allocation-mode per-vrf
  interface nve $nveId
```

```
    member vni $include_vrfSegmentId associate-vrf
configure terminal
```

Note * If the core-facing WAN uses the same RT value, add the following route-targets to simplify the manual configuration.

```
      route-target import $include_client_import_ipv4_bgpRT_1
      route-target export $include_client_export_ipv4_bgpRT_1
      route-target import $include_client_import_ipv4_bgpRT_2
      route-target export $include_client_export_ipv4_bgpRT_2
      .......
# MPLS L3VPN Universal profile
configure profile defaultNetworkMplsL3vpnDcProfile
  ipp tenant $vrfName $client_id
  include profile any
end


# MT Full VRF tenant profile
configure profile vrf-tenant-profile
  vni $vrfSegmentId
  bridge-domain $vrfBridgeDomainId
    member vni $vrfSegmentId
  interface bdi $vrfBridgeDomainId
    vrf member $vrfName
    ip forward
    no ip redirects
    ipv6 forward
    no ipv6 redirects
    no shutdown
end
```

3) Instantiating an OpFlex framework to an ACI fabric is manually configured as follows:

```
# Enable IPP feature
feature ipp

# Add ipp owned global configuration
ipp
  profile-map profile defaultNetworkMplsL3vpnDcProfile include-profile
vrf-common-mpls-l3vpn-dc-edge
  local-vtep nve 1
  bgp-as 100
  identity 11.1.1.1

# Configure to allocate bridge-domain assignments
system bridge-domain 100-3000
system fabric bridge-domain 2000-3000

# Add fabric forwarding configuration for the auto-config
feature-set fabric
feature fabric forwarding

# 'border' enables bgp evpn to work
fabric forwarding switch-role dci-node border

# timers are used to cleanup and remove recovered configurations
fabric database timer cleanup 5
fabric database timer recovery 15

# DCI Setup infra connectivity to OpFlex (interfaces are fabric facing)
interface e3/1.1
  no shutdown
  encapsulation dot1q 4
  ip address 10.1.1.1/24
  ip ospf network point-to-point
  ip router ospf 100 area 0.0.0.100

# Add IPP owned per ACI/OpFlex instance configuration
ipp
```

```
 fabric 1
    opflex-peer 10.2.2.2 8009
    ssl encrypted
  fabric 2
    opflex-peer 10.2.3.2 8009
    ssl encrypted
```

Note Default port for connecting to the OpFlex proxy server on the spine is 8009.

4) Per-tenant configuration

In a scenario, where the DCI is connected to multiple ACI PODs, and has an OpFlex framework to each POD, spines in each POD will send a tenant policy update with the BGP RT value used by that POD. DCI will add each RT as an import/export RT for the tenant VRF in question.

Following per-tenant configuration will be auto-generated on receiving fabric tenant ADD event for the first time:

```
vrf context cust1
  vni 10000
  rd auto
  address-family ipv4 unicast
    route-target import 1000:1000 evpn
    route-target export 1000:1000 evpn
  address-family ipv6 unicast
    route-target import 1000:1000 evpn
    route-target export 1000:1000 evpn

router bgp 65000
  vrf cust1
    address-family ipv4 unicast
      advertise l2vpn evpn
    address-family ipv6 unicast
      advertise l2vpn evpn

interface nve 1
  member vni 10000 associate-vrf

vni 10000
bridge-domain 100
  member vni 10000
interface bdi 100
  vrf member cust1
  ip forward
  no ip redirects
  ipv6 forward
  no ipv6 redirects
  no shutdown

ipp tenant cust1 1
```

Show and Debug Command Examples

The following examples list the show and debug commands that are used in IPP configuration.

```
switch# show ipp internal ?
  event-history  Show various event logs of IPP
  mem-stats       Dynamic memory stats
  pss                  Internal IPP pss info
  work-info       Internal IPP worker thread info
switch# show ipp internal debug
IPP Debug information
Debug Flags                    : Off
```

```
Debug-filters                  : Off

switch# show ipp internal event-history ?
  errors    Show error logs of IPP
  events    Show IPP process events
  ha        Show IPP process HA events
  msgs      Show various message logs of IPP
  opflex    Show IPP process opflex events
  periodic  Show IPP process periodic events
  trace     Show processing logs of IPP
switch# show ipp internal event-history ha
Process Event logs of IPP
2016 May 27 17:06:50.843014 ipp [6452]: [6492]: Updated tenant instance 1 rd coke-13:coke-13
 from PSS
2016 May 27 17:06:50.842880 ipp [6452]: [6492]: Updating tenant in PSS TLVU
2016 May 27 17:06:48.606305 ipp [6452]: [6492]: Updated tenant instance 1 rd coke-10:coke-10
 from PSS
2016 May 27 17:06:48.606168 ipp [6452]: [6492]: Updating tenant in PSS TLVU
2016 May 27 17:06:47.245350 ipp [6452]: [6492]: Updated tenant instance 1 rd coke-72:coke-72
 from PSS
2016 May 27 17:06:47.245169 ipp [6452]: [6492]: Updating tenant in PSS TLVU
2016 May 27 17:06:46.377087 ipp [6452]: [6492]: Updated tenant instance 1 rd coke-63:coke-63
 from PSS
2016 May 27 17:06:46.376936 ipp [6452]: [6492]: Updating tenant in PSS TLVU
2016 May 27 17:06:45.699181 ipp [6452]: [6492]: Updated tenant instance 1 rd coke-28:coke-28
 from PSS
2016 May 27 17:06:45.698969 ipp [6452]: [6492]: Updating tenant in PSS TLVU
2016 May 27 17:06:45.008724 ipp [6452]: [6492]: Updated tenant instance 1 rd coke-45:coke-45
 from PSS
2016 May 27 17:06:45.008545 ipp [6452]: [6492]: Updating tenant in PSS TLVU
2016 May 27 17:06:44.413233 ipp [6452]: [6492]: Updated tenant instance 1 rd coke-19:coke-19
 from PSS
2016 May 27 17:06:44.413075 ipp [6452]: [6492]: Updating tenant in PSS TLVU
.....................

switch# show ipp internal event-history events
Process Event logs of IPP
2016 May 27 17:08:16.457294 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-99
218 command
2016 May 27 17:08:15.105985 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-98
235 command
2016 May 27 17:08:14.370121 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-97
216 command
2016 May 27 17:08:13.133061 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-96
198 command
2016 May 27 17:08:12.331485 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-95
201 command
2016 May 27 17:08:11.052111 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-94
209 command
2016 May 27 17:08:10.341556 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-93
173 command
2016 May 27 17:08:09.061573 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-92
184 command
2016 May 27 17:08:08.376121 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-91
255 command
2016 May 27 17:08:07.183026 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-90
260 command
2016 May 27 17:08:06.483588 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-9 2
.....................
switch# show ipp internal event-history msgs
Msg events for IPP Process
1) Event:E_DEBUG, length:45, at 581444 usecs after Mon May 30 11:34:23 2016
    [100] [19461]: nvdb: transient thread created

2) Event:E_DEBUG, length:82, at 579664 usecs after Mon May 30 11:34:23 2016
    [100] [6495]: comp-mts-rx opc - from sap 19164 cmd ipp_show_internal_event_h
ist_cmd
3) Event:E_DEBUG, length:49, at 882139 usecs after Mon May 30 11:33:45 2016
    [100] [19410]: nvdb: terminate transaction failed
.....................................
switch# show ipp internal event-history opflex
Process opflex logs of IPP
```

```
2016 May 30 11:05:46.967301 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-72/GbpRoutingDomain/coke-72/
2016 May 30 11:05:46.967242 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-63/GbpRoutingDomain/coke-63/
2016 May 30 11:05:46.967165 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-28/GbpRoutingDomain/coke-28/
2016 May 30 11:05:46.716185 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-45/GbpRoutingDomain/coke-45/
2016 May 30 11:05:46.715810 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-19/GbpRoutingDomain/coke-19/
2016 May 30 11:05:46.715754 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-90/GbpRoutingDomain/coke-90/
2016 May 30 11:05:46.715696 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-73/GbpRoutingDomain/coke-73/
2016 May 30 11:05:46.715639 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-82/GbpRoutingDomain/coke-82/
2016 May 30 11:05:46.715580 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-55/GbpRoutingDomain/coke-55/
2016 May 30 11:05:46.715214 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-29/GbpRoutingDomain/coke-29/
2016 May 30 11:05:46.715153 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
..............................................
```

```
switch# show ipp internal event-history periodic
Process periodic event logs of IPP
2016 May 27 17:05:28.931685 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.927410 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.924043 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.383726 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.380290 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.376599 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.373088 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.368292 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.364850 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.361421 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.357986 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.354585 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.351387 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.347969 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.343087 ipp [6452]: [6493]: IPP Worker thread in progress
.....................................
```

```
switch# show ipp internal event-history trace
Trace logs of IPP
2016 May 30 01:07:32.962911 ipp [6452]: [6492]: sysmgr consumed mts[2558] messag
e MTS_OPC_SYSMGR_PSS_SHRINK, drop
2016 May 30 01:07:32.962893 ipp [6452]: [6492]: sysmgr processing mts[2558] mess
age MTS_OPC_SYSMGR_PSS_SHRINK
2016 May 28 01:07:29.561840 ipp [6452]: [6492]: sysmgr consumed mts[2558] messag
e MTS_OPC_SYSMGR_PSS_SHRINK, drop
2016 May 28 01:07:29.561818 ipp [6452]: [6492]: sysmgr processing mts[2558] mess
age MTS_OPC_SYSMGR_PSS_SHRINK
2016 May 27 17:06:50.843023 ipp [6452]: [6492]: Done processing MTS[250880] mess
sage MTS_OPC_HMM dropped
.................
```

```
switch# show ipp internal mem-stats
Mem stats for IPP Process
Num blocks     User size     Total size    Library
--------------------------------------------------------------------------------
     304           155244        162128  ipp
     332            64567         70336  ld-2.15.so
     120             5712          8504  libc-2.15.so
       3               60           120  libdl-2.15.so
       1                8            24  librt-2.15.so
   22877          1053813       1533896  libstdc++.so.6.0.16
      28             2294          2872  libvlan_mgr.so.0.0.0
      12             2860          3144  libsviifdb.so.0.0.0
       1              408           432  libltlmap.so.0.0.0
       9              576           792  libindxobj.so.0.0.0
     230             6472         11072  libast_db.so.0.0.0
     116             2304          4648  libavl.so.0.0.0
```

```
       4              8256          8352  libxos_lpss.so.0.0.0
      14              2604          2928  libfsrv.so.0.0.0
    3544            148784        193384  libcrypto.so.1.0.0
      99              6702          8728  libsdb.so.0.0.0
     358             11340         18544  libglib-2.0.so.0.3200.3
       2                80           128  libstartup-config.so.0.0.0
      32            151948        152648  libsdwrap.so.0.0.0
     162              5952          9296  libfsrv_client.so.0.0.0
       1               256           280  libcmd.so.0.0.0
      94              6195          8280  libcli_common.so.0.0.0
      28             31332         31960  librsw.so.0.0.0
       5             41120         41232  libtcp_clt.so.0.0.0
       3               152           216  libutils.so.0.0.0
      88             24520         26680  libpss.so.0.0.0
       3               528           600  libmts.so.0.0.0
       9              1800          2016  libsysmgr.so.0.0.0
       2             65804         65848  libopflex.so.0.0.0
       2               512           560  libuv.so.0.0.0
     689             34556         51008  libmtrack.so.0.0.0

 Grand totals:
        29172 Blocks
        1836759 User Bytes
        2420656 Total Bytes
```

```
switch# show ipp internal pss
-------------------------------
PSS Data
-------------------------------

IPP_PSS_KEY_INIT_STATE
    Server state       : 1
IPP_PSS_KEY_GENINFO
    Global tenant id   : 365
IPP_PSS_KEY_TENANT
    Fabric id              : 1
    Fabric vrf name        : coke-1:coke-1
    Vrf name               : dci_coke-1
    Tenant id              : 289
    Hmm hostid             : 289
    V4 RT (import/export)  : 1:1/1:1
    V6 RT (import/export)  : 1:1/1:1
    Flags                  : 0x0
IPP_PSS_KEY_TENANT
    Fabric id              : 1
    Fabric vrf name        : coke-10:coke-10
    Vrf name               : dci_coke-10
    Tenant id              : 266
    Hmm hostid             : 266
    V4 RT (import/export)  : 1:10/1:10
    V6 RT (import/export)  : 1:10/1:10
    Flags                  : 0x0
IPP_PSS_KEY_TENANT
    Fabric id              : 1
    Fabric vrf name        : coke-100:coke-100
    Vrf name               : dci_coke-100
    Tenant id              : 346
    Hmm hostid             : 346
    V4 RT (import/export)  : 1:100/1:100
    V6 RT (import/export)  : 1:100/1:100
    Flags                  : 0x0
IPP_PSS_KEY_TENANT
    Fabric id              : 1
    Fabric vrf name        : coke-11:coke-11
    Vrf name               : dci_coke-11
    Tenant id              : 351
    Hmm hostid             : 351
    V4 RT (import/export)  : 1:11/1:11
    V6 RT (import/export)  : 1:11/1:11
    Flags                  : 0x0
.....................
```

```
switch# show ipp internal vrf-db
   1: Vrf: dci_coke-1
        0: RT v4:(1:1,1:1) v6:(1:1,1:1)
           Host Id: 289, # tenants: 1
             Tenant Id: 289
   2: Vrf: dci_coke-10
        0: RT v4:(1:10,1:10) v6:(1:10,1:10)
           Host Id: 266, # tenants: 1
             Tenant Id: 266
   3: Vrf: dci_coke-100
        0: RT v4:(1:100,1:100) v6:(1:100,1:100)
           Host Id: 346, # tenants: 1
             Tenant Id: 346
   4: Vrf: dci_coke-11
        0: RT v4:(1:11,1:11) v6:(1:11,1:11)
           Host Id: 351, # tenants: 1
             Tenant Id: 351
   5: Vrf: dci_coke-12
        0: RT v4:(1:12,1:12) v6:(1:12,1:12)
           Host Id: 281, # tenants: 1
             Tenant Id: 281
   6: Vrf: dci_coke-13
        0: RT v4:(1:13,1:13) v6:(1:13,1:13)
           Host Id: 275, # tenants: 1
             Tenant Id: 275
   7: Vrf: dci_coke-14
        0: RT v4:(1:14,1:14) v6:(1:14,1:14)
           Host Id: 305, # tenants: 1
             Tenant Id: 305
.......................
switch# show ipp internal work-info
IPP Worker information

Work in Progress                        : False
Update queue size                       : 0
#Worker walk                            : 354
#Timedout work                          : 0
#Work done                              : 365
#Signal worker thread                   : 365

switch# show ipp fabric
Global info:
  config-profile defaultNetworkMplsL3vpnDcProfile
  include-config-profile vrf-common-mpls-l3vpn-dc-edge
  local-vtep nve 1
  bgp-as 100
  identity 1.1.1.1

Fabric 1 (Healthy)
  opflex-peer 20.4.11.1:8009 (Connected and ready)
  ssl encrypted

  Tenant Policies
     1: Fabric Vrf: coke-1:coke-1, Vrf: dci_coke-1
        RT v4:(1:1,1:1) v6:(1:1,1:1)
        Id 289, HostId: 289
        flags 0x0
     2: Fabric Vrf: coke-10:coke-10, Vrf: dci_coke-10
        RT v4:(1:10,1:10) v6:(1:10,1:10)
        Id 266, HostId: 266
        flags 0x0
     3: Fabric Vrf: coke-100:coke-100, Vrf: dci_coke-100
        RT v4:(1:100,1:100) v6:(1:100,1:100)
        Id 346, HostId: 346
        flags 0x0
.................
switch# show tech-support ipp
`show running-config ipp`

!Command: show running-config ipp
```

```
!Time: Wed Jun  1 08:37:18 2016

version 7.3(0)DX(1)
feature ipp

ipp
  profile-map profile defaultNetworkMplsL3vpnDcProfile include-profile vrf-commo
n-mpls-l3vpn-dc-edge
  local-vtep nve 1
  bgp-as 100
  identity 1.1.1.1
  fabric 1
    opflex-peer 20.4.11.1 8009
    ssl encrypted
ipp tenant dci_coke-1 289
ipp tenant dci_coke-10 266
ipp tenant dci_coke-100 346
ipp tenant dci_coke-11 351
ipp tenant dci_coke-12 281
ipp tenant dci_coke-13 275
ipp tenant dci_coke-14 305
ipp tenant dci_coke-15 292
ipp tenant dci_coke-16 339
ipp tenant dci_coke-17 323
ipp tenant dci_coke-18 330
ipp tenant dci_coke-19 361
ipp tenant dci_coke-2 310
ipp tenant dci_coke-20 350
ipp tenant dci_coke-21 283
ipp tenant dci_coke-22 272
ipp tenant dci_coke-23 299
ipp tenant dci_coke-24 294
ipp tenant dci_coke-25 337
ipp tenant dci_coke-26 326
ipp tenant dci_coke-27 334
ipp tenant dci_coke-28 363
ipp tenant dci_coke-29 356
ipp tenant dci_coke-3 343
ipp tenant dci_coke-30 277
ipp tenant dci_coke-31 307
ipp tenant dci_coke-32 293
ipp tenant dci_coke-33 341
ipp tenant dci_coke-34 321
ipp tenant dci_coke-35 329
ipp tenant dci_coke-36 269
ipp tenant dci_coke-37 352
ipp tenant dci_coke-38 285
.............
switch# debug ipp ?
  all         All debugs
  cli         CLI command processing debugs
  event     IPP events
  ha          HA related debugs
  hmm       IPP HMM api debug
  opflex    IPP opflex debugs
  periodic IPP events periodic
```

Feature History for ACI WAN Interconnect

This table lists the release history for this feature.

Table 1: Feature History for ACI WAN Interconnect

Feature Name	Releases	Feature Information	
ACI WAN Interconnect	7.3(1)D1(1)	This feature was introduced.	