Cisco Programmable Fabric with VXLAN BGP EVPN Configuration Guide

First Published: 2016-04-28

Last Modified: 2019-08-27

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

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

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

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.

CONTENTS

PREFACE

CHAPTER 1

CHAPTER 2

CHAPTER 3

Preface

The Preface contains the following sections:

Audience

This publication is for experienced network administrators who configure and maintain Cisco Programmable Fabric.

Document Conventions

Command descriptions use the following conventions:

Note As part of our constant endeavor to remodel our documents to meet our customers' requirements, we have modified the manner in which we document configuration tasks. As a result of this, you may find a deviation in the style used to describe these tasks, with the newly included sections of the document following the new format.

Convention	Description
bold	Bold text indicates the commands and keywords that you enter literally as shown.
Italic	Italic text indicates arguments for which the user supplies the values.
[x]	Square brackets enclose an optional element (keyword or argument).
[x | y]	Square brackets enclosing keywords or arguments separated by a vertical bar indicate an optional choice.

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

Examples use the following conventions:

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

This document uses the following conventions:

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the manual.

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Related Documentation for Cisco Programmable Fabric

Software Downloads, Release, and General Information

Cisco Programmable Fabric Release Notes:

http://www.cisco.com/c/en/us/td/docs/switches/datacenter/pf/release_notes/programmable_fabric_rel_notes.html

Cisco DCNM Release Notes, Release 10:

http://www.cisco.com/c/en/us/td/docs/switches/datacenter/sw/10_0_x/release_notes/b_dcnm_release_notes_10_0.html

Install and Upgrade Guides

Cisco DCNM 10 Installation Guide:

http://www.cisco.com/c/en/us/td/docs/switches/datacenter/sw/10_0_x/installation/DCNM_Installation_Guide_10_0_x.html

Configuration Guides

Cisco Programmable Fabric Configuration Guide:

http://www.cisco.com/c/en/us/td/docs/switches/datacenter/pf/configuration/guide/b-pf-configuration.html

Cisco DCNM 10 Fundamentals Guide:

http://www.cisco.com/c/en/us/td/docs/switches/datacenter/sw/10_0_x/fundamentals/DCNM_Fundamentals_10.html

Cisco Nexus 1000V VDP Configuration Guide, Release 5.x:

http://www.cisco.com/c/en/us/support/switches/nexus-1000v-switch-vmware-vsphere/products-installation-and-configuration-guides-list.html

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see *What's New in Cisco Product Documentation*, at: http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html.

Subscribe to *What's New in Cisco Product Documentation*, which lists all new and revised Cisco technical documentation, as an RSS feed and deliver content directly to your desktop using a reader application. The RSS feeds are a free service.

CHAPTER 1

Introduction to Cisco Programmable Fabric

Introduction to Programmable Fabric

Licensing Requirements

For a complete explanation of Cisco NX-OS licensing recommendations and how to obtain and apply licenses, see the *Cisco NX-OS Licensing Guide*.

Fabric Technology Overview

Data Center Network Evolution

The Data Center fabric journey evolved from a Spanning-Tree (STP) based network to more efficient ways of using the available resources of the infrastructure.

Initially, we built Spanning-Tree (STP) based networks where a single link was active for a given service (VLAN). The same was true for the demarcation between Layer 2 and Layer 3, where a First Hop Redundancy Protocol like HSRP or VRRP provided a single active point of egress per service (VLAN/subnet).

With the introduction of Cisco's version of Multi-Chassis Link-Aggregation, virtual Port-Channels (vPC), a significant improvement has been achieved by providing a loop-free topology. In vPC-based environments, Spanning-Tree (STP) was still present to provide a failsafe mechanism without the disadvantages of a single path tree active. vPC also provided enhancements to the Layer 2 / Layer 3 demarcation, where now First Hop Redundancy Protocols (FHRP) like HSRP and VRRP start forwarding in an active/active manner, and this improved a former chokepoint. Nevertheless, even with vPC and FHRP active/active, further scale-out with regards to Layer 2 and Layer 3 forwarding became an ask.

Cisco introduced Layer 2 Multipathing (L2MP) to accommodate these asks with the introduction of Cisco FabricPath. With the MAC-in-MAC frame encapsulation and the IS-IS routing protocol, Cisco provided a Layer 2 Equal Cost Multipath (ECMP) based network, where hosts were allowed to talk to other hosts across all available links. Different to vPC, FabricPath did not require a pairing of network nodes and the configuration became simplified. A Layer 2 fabric was made available and the first scale-out network architecture was embraced; the need for a wider Layer 2 / Layer 3 demarcation became eminent. With Anycast-HSRP, Cisco implemented a way to scale-out the common chokepoint for First-Hop gateways and extended it to 4 active nodes. Further scale-out was an ask for FabricPath and enhancements have been added like Enhanced

Forwarding, Distributed Anycast Gateway at the Leaf switch, and automation of connected workloads; these enhancements fell under Cisco's Dynamic Fabric Automation (DFA) solution.

With the industry moving from Frame Encapsulation (MAC-in-MAC) to Packet Encapsulation (MAC-in-IP), Cisco embraced VXLAN within its Data Center Switching portfolio to provide a standards based encapsulation technique. Initially, VXLAN was introduced as a Layer 2 service only and since VXLAN Flood and Learn (as defined in RFC 7348) follows the similar Flood and Learn semantic as Ethernet or FabricPath, enhancements were required. VXLAN with a control plane became necessary to introduce Layer 2 and Layer 3 services, while optimizing forwarding to address the limitations of VXLAN Flood and Learn. Multiprotocol Border Gateway Protocol with Ethernet Virtual Private Network (MP-BGP EVPN) was introduced as the control plane, with VXLAN being used in the data plane. MP-BGP EVPN has been defined by IETF as the standards-based control plane for VXLAN overlays. The Programmable Fabric solution is based on VXLAN with BGP EVPN, with the programmability of the network fabric through APIs.

Data Center fabric challenges remained present and included disjoint provisioning, CLI centric box-by-box configuration, disruptive growth of applications (and other data center entities), deficient host overlay, and location dependency (rigid coupling of IP address to location). These challenges have led to operational complexity, architectural rigidness and infrastructure inefficiency.

The Programmable Fabric solution addresses the above challenges by providing the following:

- A programmable infrastructure.
- An open set of APIs.
- Scale-out architecture.
- Hybrid overlays.
- Total host and IP mobility (decoupling the identity of a host to its location).

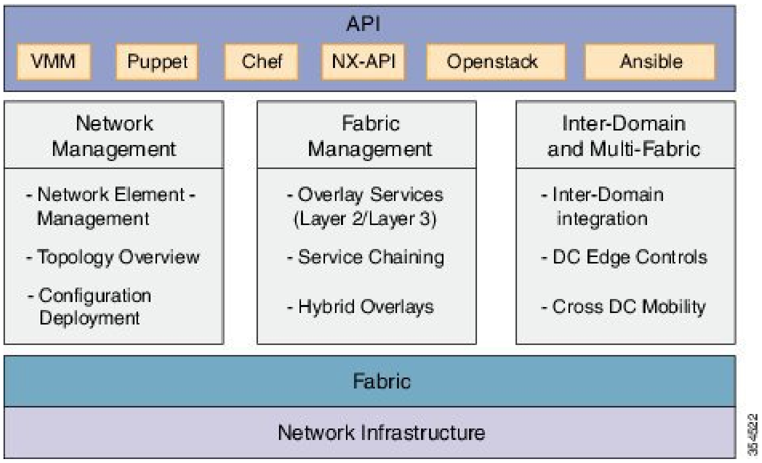
This results in simplifying of the underlying fabric and optimizing the overlay for north-south as well as east-west traffic flows, which eases the placement of workload as the network scales with integrated automation

Understanding Programmable Fabric (with VXLAN BGP EVPN)

The Programmable Fabric comprises of the Cisco Nexus 2000, 5000, 7000, and 9000 Series switches, VXLAN BGP EVPN implementation on the platform, and a provision for APIs atop this infrastructure.

Note The Cisco Nexus Series switches provide VXLAN bridging and routing in hardware.

Figure 1: Programmable Fabric overview



Considerations for building the Programmable Fabric are given below:

- *The underlay* — Considerations for building the underlay include choice of IP protocol, topology, etc.
- *The overlay* — Layer 2 and Layer 3 overlay configuration and monitoring.
- *Hybrid overlays* — Integration of physical and virtual VTEPs.
- *Inter domain and multi fabric function* — Interconnection of different data center pods and data center sites in a secure, scalable manner and optimizing traffic flow between them.

APIs — On top of the above underlying infrastructure, the Programmable Fabric solution allows implementation of an open set of APIs (such as NX-API, Chef, Puppet, OpenStack, etc) to automate and orchestrate the Programmable Fabric.

Programmable Fabric Properties

General properties

- *Any subnet, anywhere* provision — A tenant's servers can be attached to any Top of Rack (ToR)/leaf switch in the VXLAN cluster and still be recognized as part of a specific tenant network.
- *Reduced failure domains* — Failure domains are restricted to the leaf layer or ToR, or to a pair of leaf switches. A reduction in failure domains increases the reliability of the datacenter network.
- *Extensible scale and resiliency* —Availability of full bisectional bandwidth and multiple paths to any given destination.
- *Profile controlled configuration* — Centralized controllers help in automated provisioning of devices and configurations. This reduces human errors and significantly simplifies configuration. This is done through an orchestrator.

Spine/Leaf Topologies

- Availability of high bisectional bandwidth due to the nature of the leaf switch to spine switch interconnectivity in the fabric. The leaf-spine fabric is (mathematically) the best way of communication between ingress and egress switches in a given network.

Note For additional information on CLOS fabric, refer to resources such as Wikipedia.

- More spine switches provide increased bandwidth and resiliency, and more leaf switches provide more ports and capacity.
- Leverages Layer 3 ECMP for unicast or multicast traffic. All links are used for forwarding traffic.
- Uniform reachability across the network and deterministic latency. No HSRP is required since each leaf switch has a distributed anycast gateway.
- High redundancy is available for node/link failure in the fabric. You can use a vPC setup for redundancy between two leaf or border leaf switches.
- Line rate, consistent latency, for all traffic – You will have the same number of hops between any two ToRs.
- Flexibility as regards to subscription ratio – You can change the subscription ratio between spine and leaf switches as the fabric scales up or decreases in size.
- You can use a two stage or multi stage leaf/spine switch topology for connecting two DC pods.

Variety of south bound topological connectivity (at the leaf switch level)

You can choose the appropriate mix of options for your organization. Some options are given below. They are explained in detail in the next sub section.

- FEX in a straight through or dual active mode (eVPC).

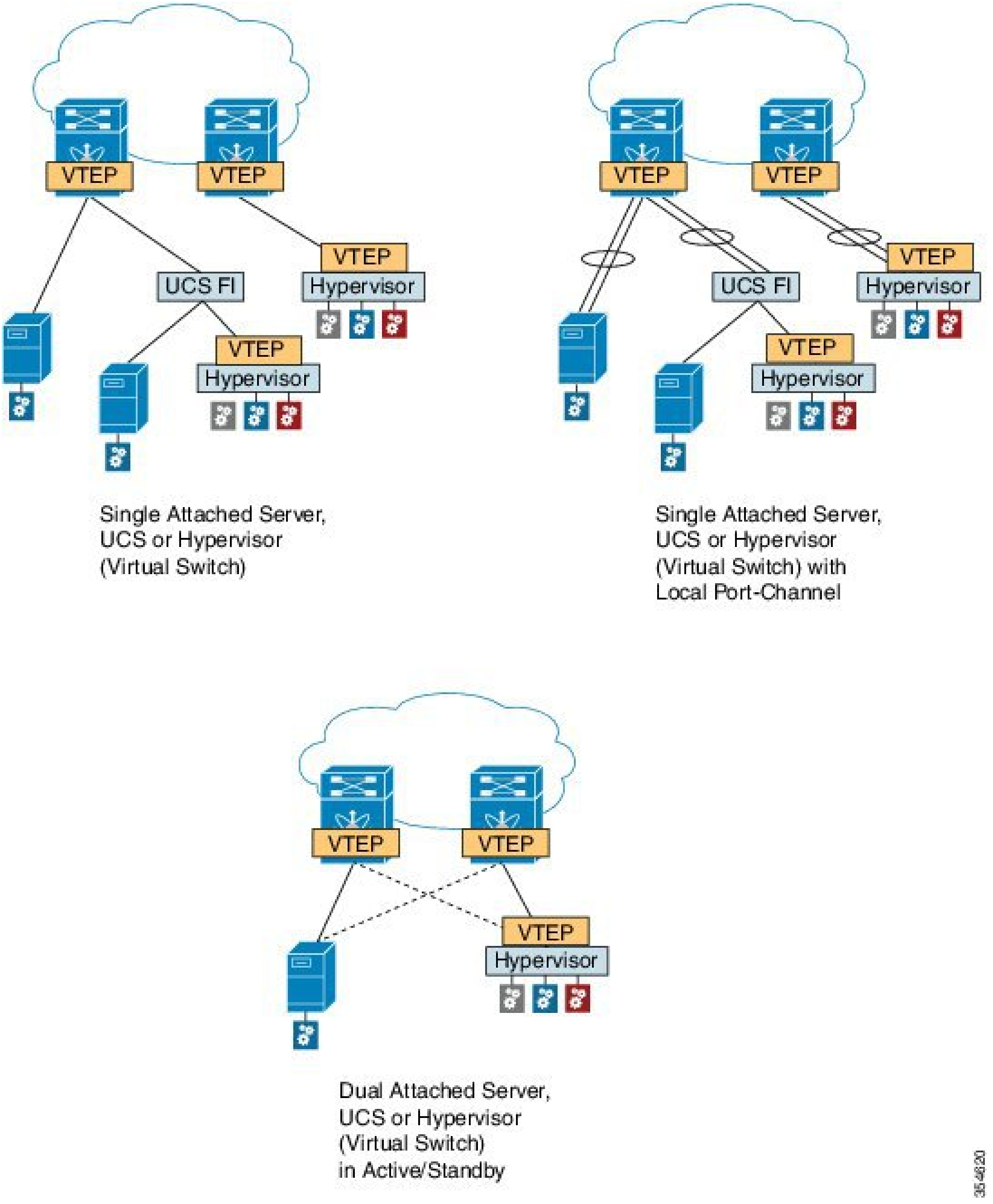
Note When implementing FEX, ensure that you are aware of the hardware and software dependencies.

- Blade switches.
- Hypervisors or bare metal servers attached in vPC mode.
- Active-Standby and Active-Active connectivity options.
- Connecting to external networks through border spine or border leaf switches.
- Services can be attached to service nodes.

Connectivity Options and End-Host attachment

Southbound connectivity options are given below. Choose the appropriate option for your setup.

Figure 2: Southbound connectivity options

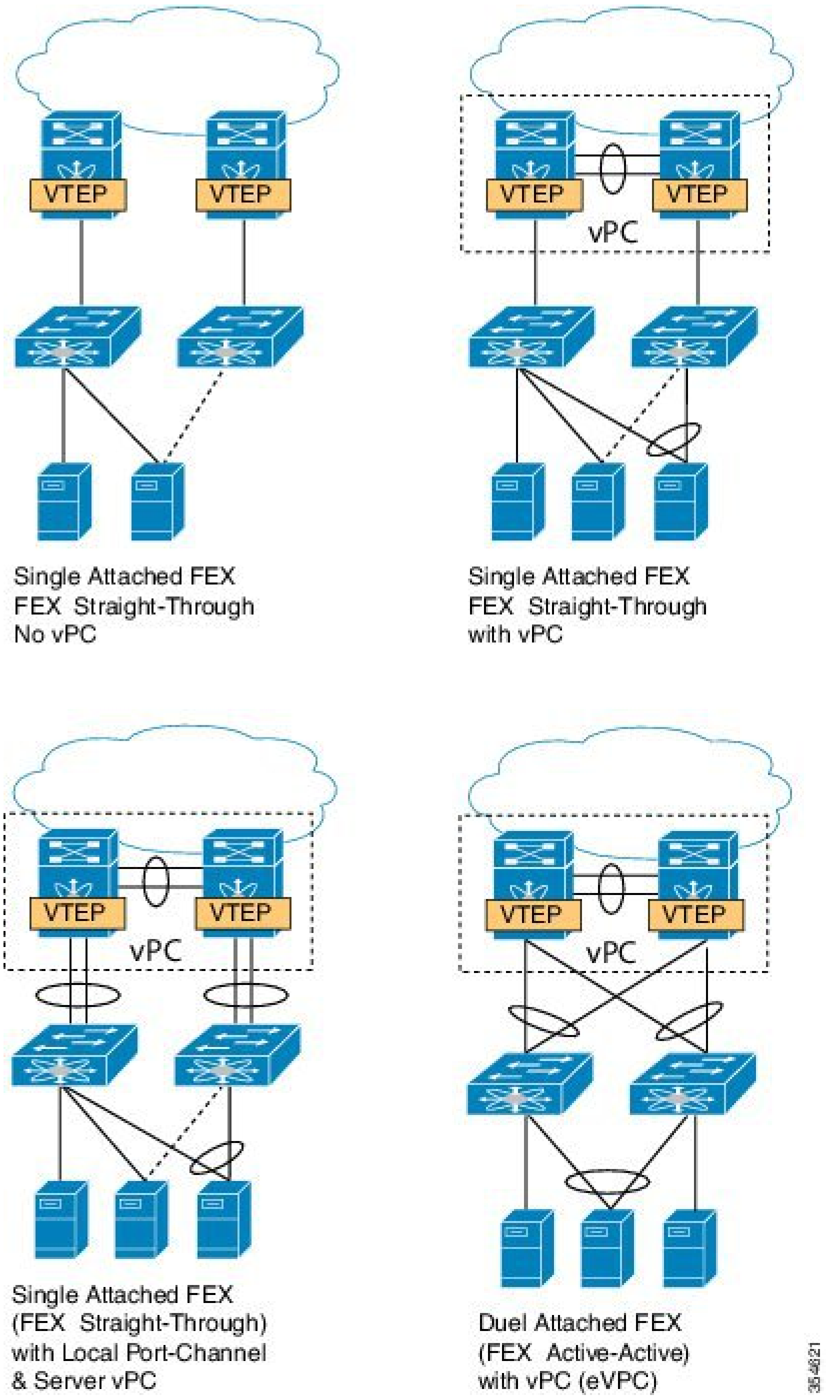


- Option 1 —Single threaded links option - In the example, a physical server, UCS FI, and a hypervisor for VMs are connected as single threaded links.
- Option 2 —Single threaded link + a local port channel (through an LACP connection) for each link, for physical layer redundancy.
- Option 3 —A dual attached physical server and hypervisor in an Active-Standby mode for redundancy.

Southbound connectivity for servers with FEX

If the above options are used with FEXs that are attached to the ToR switches, then the topologies are changed, as explained below.

Figure 3: Southbound connectivity with FEX



- Option 1 —A single attached FEX. This is not recommended.

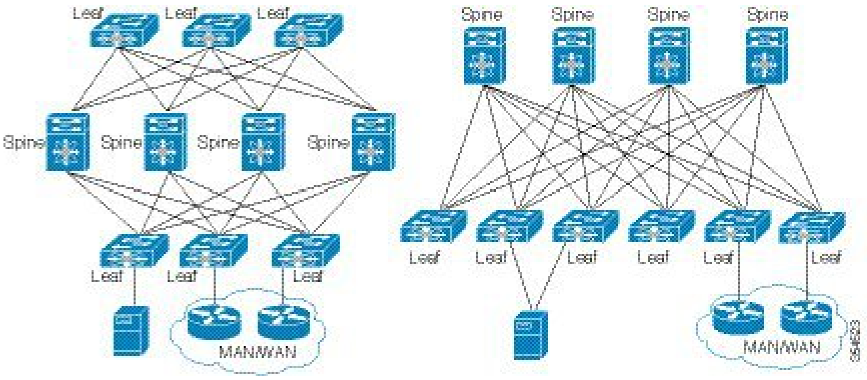
- Option 2 —A single attached FEX scenario with vPC enabled at the leaf switch level. One host attached to the FEX is single attached, another an Active-Standby host, and the third host has an Active-Active connection via a vPC port channel.
- Option 3 —This is the same as the second option with one addition. The FEX and the leaf switch devices are connected though a port channel for link redundancy.
- Option 4 —A dual attached FEX scenario for FEX and leaf switch redundancy. This option gives more resiliency for the single attached host to reach one of the leaf switches. The second host is dual attached with vPC.

Roles and Terminology

General properties

A CLOS based Spine-Leaf architecture is used for the data center fabric in the Programmable Fabric solution. The example displays a folded, 2-stage CLOS Spine-Leaf fabric topology to the left and unfolded representation of the same topology to the right. Note that both the topologies use the same number of devices and links.

Figure 4: Folded and unfolded CLOS topology



The various components and their roles are described below:

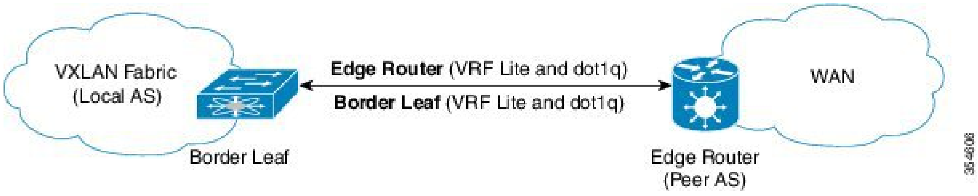
- Spine switch
  - IP transport forwarder between Leaf switches (East-West).
  - Potentially hosting Rendezvous-Point (RP) for the underlay.
  - Potentially hosting BGP Route-Reflector (RR) for EVPN.
  - Does not require the VTEP functionality.
  - Interconnects leaf switches and border leaf switches.
- Leaf switch – (Also referred as a ToR switch or Edge-device hosting the VTEP.)
  - VXLAN edge device to which end hosts are attached. The end hosts include virtual and physical/bare metal servers, FEX devices, 3rd party switches, UCS FI, controllers, and blade switches.
  - Routes or bridges Classic Ethernet frames and encapsulates them into VXLAN.
  - Requires the VTEP functionality.

- Border Leaf/Spine switch for external connectivity
    - VXLAN edge device.
    - Routes and bridges Classical Ethernet frames from an outside network and encapsulates them into VXLAN (North-South).
    - Decapsulates MPLS PE/LISP traffic from an outside network and re-encapsulates it into VXLAN (North-South).
    - Speaks IGP/EGP routing protocols with the outside network (North-South).
    - Requires the VTEP functionality.
    - Interface options are physical routed ports, sub interfaces, and VLAN SVIs over trunk ports.
    - IPv4/IPv6 route exchange with external neighbors.
- Border Spine switch – Additional functions unique to the border spine switch are given below:
    - IP transport forwarder between Leaf switches (East-West).
    - Potentially hosting Rendezvous-Point (RP) for the underlay.
    - Potentially hosting BGP Route-Reflector (RR) for EVPN.
- Services Leaf switch – (This option does not need a dedicated pair of leaf switches, and could be any combination of a leaf switch VTEP and a next-hop router).
    - Firewalls.
    - Load balancers.
    - Proxy services.
    - IPS services.

One box and two box solutions

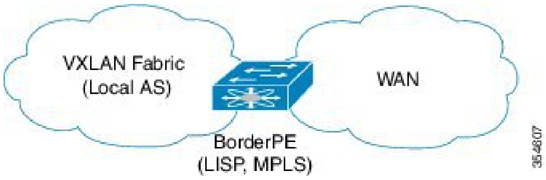
A *two box solution* comprises of two switches (A Cisco Nexus 5600/7000/7700/9000 Series border leaf switch + a WAN edge switch) to route IP frames from an external network into the VXLAN EVPN fabric.

Figure 5: Two box solution


VXLAN Fabric
(Local AS)
Border Leaf
Edge Router (VRF Lite and dot1q)
Border Leaf (VRF Lite and dot1q)
WAN
Edge Router
(Peer AS)
354606

An *one box solution* is a single Border PE switch (a Cisco Nexus 7000/7700 Series switch with an F3 line card at the spine or leaf layer) with MPLS PE function. You can connect to external networks through MPLS as well as LISP.

Figure 6: One box solution



CHAPTER 2

Introducing Cisco Programmable Fabric (VXLAN/EVPN)

Introduction to VXLAN/EVPN

Introducing IP Fabric Overlays (VXLAN)

Motivation for an overlay

An overlay is a dynamic tunnel that transports frames between two endpoints. In a switch-based overlay, the architecture provides flexibility for spine switches and leaf switches.

- Spine switch table sizes do *not* increase proportionately when end hosts (physical servers and VMs) are added to the leaf switches.
- The number of networks/tenants that can be supported in the cluster can be increased by just adding more leaf switches.

How this is achieved is explained in detail later.

Note For easier reference, some common references are explained below:

- *End host* or *server* refers to a physical or virtual workload that is attached to a ToR switch.
- A *ToR* switch is also referred as a *leaf* switch. Since the VTEP functionality is implemented on the ToRs, a VTEP refers to a ToR or leaf switch enabled with the VTEP function. Note that the VTEP functionality is enabled on all leaf switches in the VXLAN fabric and on border leaf/spine switches.

VXLAN as the overlay technology for the Programmable Fabric solution

VXLAN is a MAC in IP/UDP overlay that allows layer 2 segments to be stretched across an IP core. All the benefits of layer 3 topologies are thereby available with VXLAN including the popular layer-3 ECMP feature for efficient traffic spread across multiple available paths. The encapsulation and decapsulation of VXLAN headers is handled by a functionality embedded in VXLAN Tunnel End Points (VTEPs). VTEPs themselves could be implemented in software or a hardware form-factor.

VXLAN natively operates on a flood-n-learn mechanism where BU (Broadcast, Unknown Unicast) traffic in a given VXLAN network is sent over the IP core to every VTEP that has membership in that network. There are two ways to send such traffic: (1) Using IP multicast (2) Using Ingress Replication or Head-end Replication. The receiving VTEPs will decapsulate the packet, and based on the inner frame perform layer-2 MAC learning. The inner SMAC is learnt against the outer Source IP Address (SIP) corresponding to the source VTEP. In this way, reverse traffic can be unicasted toward the previously learnt end host.

Other motivations include:

1. *Scalability* — VXLAN provides Layer-2 connectivity that allows the infrastructure that can scale to 16 million tenant networks. It overcomes the 4094-segment limitation of VLANs. This is necessary to address today’s multi-tenant cloud requirements.
2. *Flexibility*— VXLAN allows workloads to be placed anywhere, along with the traffic separation required in a multi-tenant environment. The traffic separation is done using network segmentation (segment IDs or virtual network identifiers [VNIs]).

   Workloads for a tenant can be distributed across different physical devices (since workloads are added as the need arises, into available server space) but the workloads are identified by the same layer 2 or layer 3 VNI as the case may be.

3. *Mobility*— You can move VMs from one data center location to another without updating spine switch tables. This is because entities within the same tenant network in a VXLAN/EVPN fabric setup retain the same segment ID, regardless of their location.

Overlay example:

The example below shows why spine switch table sizes are not increased due to VXLAN fabric overlay, making them lean.

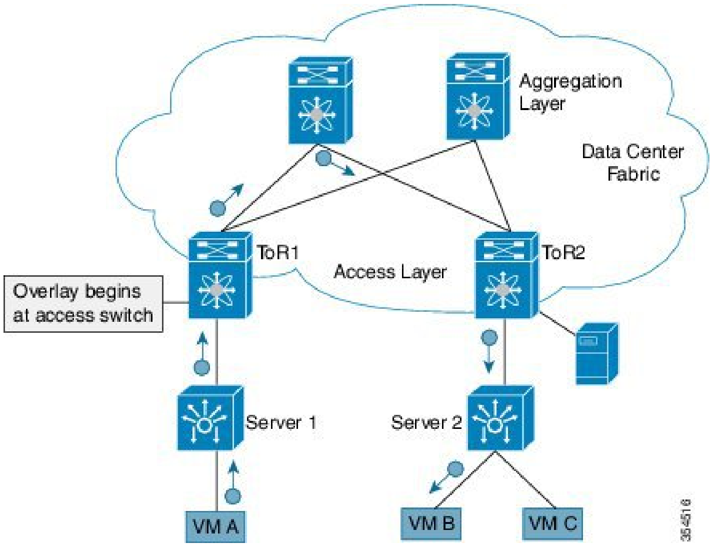
VM A sends a message to VM B (they both belong to the same tenant network and have the same segment VNI). ToR1 recognizes that the source end host corresponds to segment x, searches and identifies that the target end host (VM B) belongs to segment x too, and that VM B is attached to ToR2. Note that typically the communication between VM A and VM B belonging to the same subnet would first entail ARP resolution.

ToR1 encapsulates the frame in a VXLAN packet, and sends it in the direction of ToR2.

The devices in the path between ToR1 to ToR2 are not aware of the original frame and route/switch the packet to ToR2. .

ToR2 decapsulates the VXLAN packet addressed to it. It does a lookup on the inner frame. Through its end host database, ToR2 recognizes that VM B is attached to it and belongs to segment x, forwards the original frame to VM B.

Figure 7: VXLAN Overlay



- VXLAN semantics are in operation from ToR1 to ToR2 through the encapsulation and decapsulation at source and destination VTEPs, respectively. The *overlay* operation ensures that the original frame/packet content is not exposed to the underlying IP network.

- The IP network that sends packets from ToR1 to ToR 2 based on the outer packet source and destination address forms the *underlay* operation. As per design, none of the spine switches need to learn the addresses of end hosts below the ToRs. So, learning of hundreds of thousands of end host IP addresses by the spine switches is avoided.

Learning of (hundreds of thousands of) end host IP and MAC addresses

One of the biggest limitations of VXLAN flood-n-learn is the inherent flooding that is required ensuring that learning happens at the VTEPs. In a traditional deployment, a layer-2 segment is represented with a VLAN that comprises a broadcast domain, which also scopes BU traffic. With VXLAN, now the layer-2 segment spans a much larger boundary across an IP core where floods are translated to IP multicast (or HER). Consequently, the flood-n-learn based scheme presents serious scale challenges especially as the number of end hosts go up. This is addressed via learning using a control-plane for distribution of end host addresses. The control plane of choice is MP-BGP EVPN. By implementing MP-BGP EVPN with VXLAN, the following is made possible:

- End hosts' information is available to the attached ToR via First Hop Protocols such as ARP/ND/DHCP etc., when a new bare-metal server or VM is attached.

- End host to ToR mapping information for each ToR is shared with every other ToR using BGP via a route reflector.

- Specifically, within BGP, the EVPN address family is employed to carry MAC *and* IP address information of the end hosts along with other information such as the network and tenant (aka VRF) to which they belong. This allows optimal forwarding of both layer-2 and layer-3 traffic within the fabric.

- VMs belonging to the same tenant might be many hops apart (though assigned with the same segment ID/VNI), and there might be frequent movement and addition of end hosts. When a new VM comes up or is moved between ToRs, the information is instantly updated into BGP by the detecting ToR thereby ensuring that the updated reachability information is also known to every other ToR.

- In order to accurately route/switch packets between end hosts in the data center, each participating ToR in a VXLAN cluster must be aware of the end hosts attached to it and also the end hosts attached to other ToRs, in real time.

VXLAN-EVPN fabric— The overlay protocol is VXLAN and BGP uses EVPN as the address family for communicating end host MAC and IP addresses, so the fabric is referred thus.

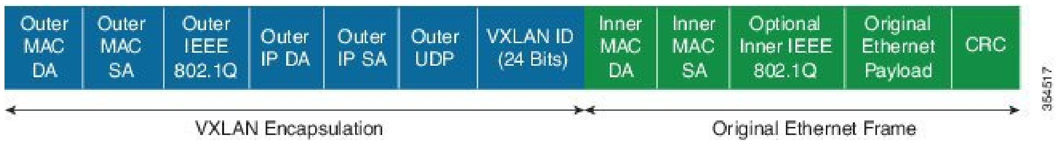
More details for MP-BGP EVPN are noted in the *Fabric Overlay Control-Plane (MP-BGP EVPN)* section

Realizing Layer-2 and Layer-3 Multi-Tenancy

Using segment IDs or VNIs for multi tenancy in the VXLAN fabric

Typically, when a tenant is created, it is assigned a unique VNI referred to as the layer-3 VNI or the layer 3 segment ID. This serves as a unique identifier for tenant layer-3 context also referred to as the tenant VRF. For each network created within the tenant, a unique identifier is assigned which is referred to as the layer-2 VNI or layer-2 segment-id. The VNIs all come from the same 2^24 – 1 pool represented by the 24-bit VNI identifier carried in the VXLAN header.

Figure 8: VXLAN Packet Format

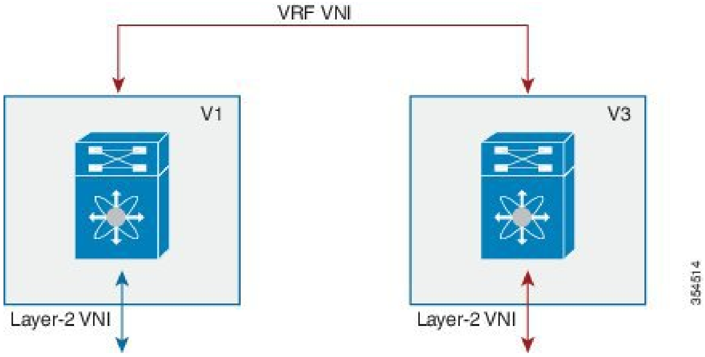


Some Segment ID/VNI pointers are given below:

- If a new VM or physical server for this tenant is added to the data center, it is associated with the *same layer-3 VNI*, regardless of the physical location. In addition, if it is part of a given tenant network, it is assigned the same layer-2 VNI that identifies that network.
- By confining server and end host identification of a specific tenant to a unique VNI (or few unique VNIs), segmentation and security are ensured.
- By ensuring that the VNI-to-end host mapping information on each ToR is updated and shared through the route reflector, the latest information is available through the VXLAN setup.
- Routing at the ToR/access layer facilitates a more scalable design, contains network failures, and enables transparent mobility.

Traffic between servers in the same tenant network that is confined to the same subnet is bridged. In this case, the VTEPs stamp the layer-2 VNI in the VXLAN header when the communication is between servers that are below different ToRs. The forwarding lookup is based on (L2-VNI, DMAC). For communications between servers that are part of the same tenant but belong to different networks, routing is employed. In this case, the layer-3 VNI is carried in the VXLAN header when communication is between servers below different ToRs. This approach is referred to as the symmetric IRB (Integrated Routing and Bridging) approach, the symmetry comes from the fact that VXLAN encapsulated routed traffic in the fabric from source to destination and vice-versa will carry the same layer-3 VNI. This is shown in the figure below.

Figure 9: Inter Tenant Traffic Flow Using VRF VNI



In the above scenario, traffic from a server (with layer-2 VNI *x*) on VTEP V1 is sent to a server (with layer-2 VNI *y*) on VTEP V2. Since the VNIs are different, the layer-3 VNI (unique to the VRF) is used for communication over VXLAN between the servers.

Fabric Overlay Control-Plane (MP-BGP EVPN)

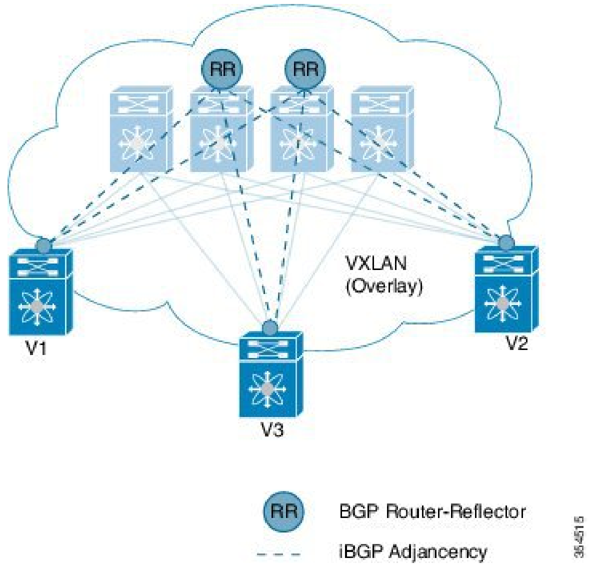
The main motivations for using BGP EVPN as the control plane are:

- *Standards based*—The overlay (VXLAN) and the control plane (BGP) are standards based.
- *Implement control-plane MAC learning* so that VMs/servers for each tenant have a unique identity across the fabric.

  In a VXLAN-EVPN based fabric, MAC learning occurs via the control plane [through multi-protocol (MP) BGP] instead of the data plane.

  When a new end host is attached to a VTEP (aka ToR), the VTEP advertises the MAC and IP address of the end host to a route reflector which in turn advertises it to the other VTEPs through MP-BGP (as shown in the image below). Since MP-BGP enables isolation of groups of interacting agents, VMs/servers that belong to the same tenant are logically isolated from other tenants.

Figure 10: End Host IP + MAC Address Distribution in a VXLAN Setup



The motivations for using BGP EVPN continues below:

- *Reduce flooding*
  - Since the number of end hosts attached to VTEPs in a data center is huge, a mechanism is required to reduce flooding for discovery of end host location and resolution information. This is achieved via MAC/IP binding information distribution.
  - MAC address distribution eliminates (or reduces) unknown unicast flooding because MAC addresses are prepopulated.
  - MAC to IP *binding* information helps in local ARP suppression.
- *Distributed anycast gateway*
  - For a given subnet, the same default gateway with the same IP and MAC address is realized simultaneously on appropriate ToR switches thereby ensuring the default gateway for the end hosts is always at its closest point aka its directly attached switch.
  - This ensures that routed traffic is also optimally forwarded within the fabric without going through any tromboning.
- *VM Mobility Support*
  - The control plane supports transparent VM mobility within and across VXLAN BGP EVPN fabrics, and quickly updates reachability information to avoid hair-pinning of east-west traffic.
  - The distributed anycast gateway also aids in supporting transparent VM mobility since post VM move, the ARP cache entry for the default gateway is still valid.
- *Efficient bandwidth utilization and resiliency with Active-Active multipathing*

  VXLAN is supported with virtual PortChannel (vPC). This allows resiliency in connectivity for servers attached to access switches with efficient utilization of available bandwidth. VXLAN with vPC is also supported for access to aggregation (leaf switch to spine switch) connectivity, promoting a highly available fabric.

- *Secure VTEPs*

  In a VXLAN-EVPN fabric, traffic is only accepted from VTEPs whose information is learnt via the BGP-EVPN control plane. Any VXLAN encapsulated traffic received from a VTEP that is not known via the control plane will be dropped. In this way, this presents a secure fabric where traffic will only be forwarded between VTEPs validated by the control plane. This is a major security hole in data-plane based VXLAN flood-n-learn environments where a rogue VTEP has the potential of bringing down the overlay network.

- *BGP specific motivations*

  - *Increased flexibility*— EVPN address family carries both Layer-2 and Layer-3 reachability information. So, you can build bridged overlays or routed overlays. While bridged overlays are simpler to deploy, routed overlays are easier to scale out.

  - *Increased security*— BGP authentication and security constructs provide more secure multi-tenancy.

  - *Improved convergence time*— BGP being a hard-state protocol is inherently non-chatty and only provides updates when there is a change. This greatly improves convergence time when network failures occur.

  - *BGP Policies*— Rich BGP policy constructs provide policy-based export and import of reachability information. It is possible to constrain route updates where they are not needed thereby realizing a more scalable fabric.

  - *Advantages of route reflectors*— Increases scalability and reduces the need for a full mesh (coverage) of BGP sessions.

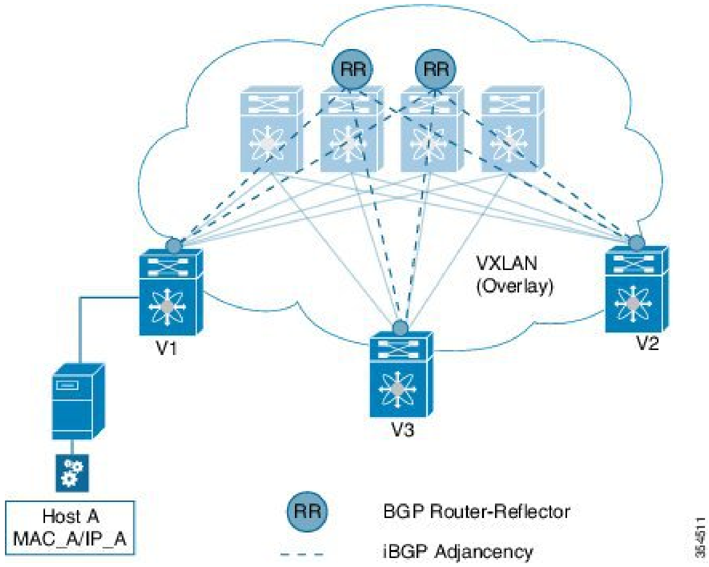
    A route reflector in an MP-BGP EVPN control plane acts as a central point for BGP sessions between VTEPs. Instead of each VTEP peering with every other VTEP, the VTEPs peer with a spine device designated as a route reflector. For redundancy purposes, an additional route reflector is designated.

End Host and Subnet Route Distribution

Some pointers about end host MAC and IP route distribution in a VXLAN EVPN fabric are given below:

- When a new end host is attached to a VTEP (say *Host A* in the below scenario), the VTEP V1 learns the end host’s MAC and IP address. MP-BGP on the VTEP nodes enables advertising of the addresses (IP + MAC) to the route reflector.

Figure 11: New End Host Attaches to a VTEP



- MP-BGP also distributes subnet routes and external reachability information between VTEPs. When VTEPs obtain end host routes of remote end hosts attached to other VTEPs, they install the routes in their RIB and FIB.

Note that the end host route distribution is decoupled from the underlay protocol.

End host communication within a VNI and across VNIs

As we know, in a VXLAN EVPN fabric a unique Layer-2 VNI is designated to each tenant network.

Recall, when two end hosts sharing the same layer-2 VNI communicate with each other, the traffic is bridged, and confined to a subnet. When two end hosts in different layer-2 VNIs communicate with each other, the traffic is routed and moves between subnets.

Furthermore, an end host retains its address and tenant association when it moves to another VTEP.

One tenant network, one Layer-2 VNI, and one default gateway IP and MAC address

Since end hosts in a tenant network might be attached to different VTEPs, the VTEPs are made to share a common gateway IP and MAC address for intra-tenant communication.

If an end host moves to a different VTEP, the gateway information remains the same and reachability information is available in the BGP control plane.

Distributed IP anycast gateway

The gateway is referred as a *distributed IP anycast gateway*, since the gateway is distributed across all relevant VTEPs.

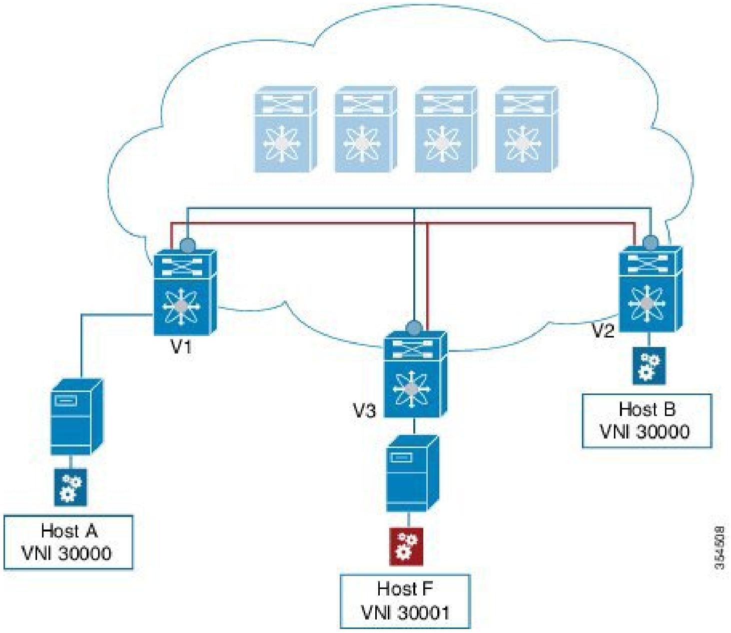
The gateway provides routing and bridging capabilities, and the mechanism is referred as *Integrated Routing and Bridging* (IRB).

The distributed anycast gateway for routing is completely stateless and does not require the exchange of protocol signalization for election or failover decision.

All VTEPs host active default gateways for their respective configured subnets and First Hop Routing Protocols (FHRP) such as HSRP, VRRP etc. are not needed.

A sample distributed gateway for a setup, and the associated configurations are given below:

Figure 12: Distributed Gateway



Blue tenant network's configuration

Similar configuration needs to be implemented on the leaf switch V1 and all other switches containing the blue tenant network's end hosts. Since VLAN configuration is local to the switch, you can configure the same VLAN ID for other switches or a different one.

VLAN to VNI mapping (MT-Lite)

(config) #

```
vlan 43
    vn-segment 30000
```

The anycast gateway MAC, inherited by any interface (SVI) using "fabric forwarding"

(config) #

```
fabric forwarding anycast-gateway-mac 2020.0000.00aa
```

Distributed IP anycast gateway (SVI)

(config) #

```
interface vlan 43
  no shutdown
  vrf member VRF-A
```

```
     ip address 10.11.11.1/24 tag 12345
     fabric forwarding mode anycast-gateway
```

Red tenant network's configuration

VLAN to VNI mapping (MT-Lite)

(config) #

```
vlan 55
  vn-segment 30001
```

The anycast gateway MAC, inherited by any interface (SVI) using “fabric forwarding”

(config) #

```
fabric forwarding anycast-gateway-mac 2020.0000.00aa
```

Distributed IP anycast gateway (SVI)

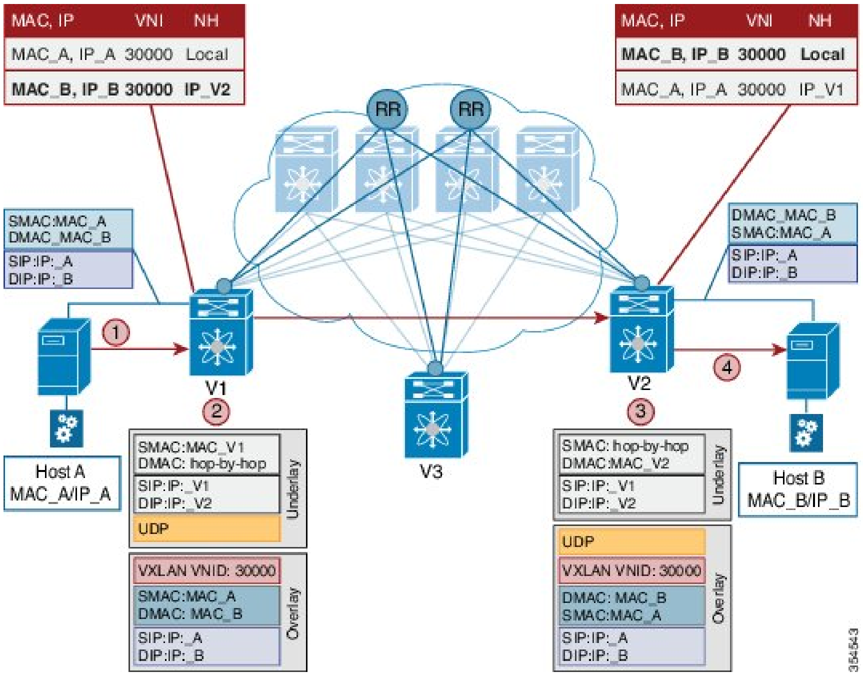
(config) #

```
interface vlan 55
  no shutdown
  vrf member VRF-A
     ip address 10.98.98.1/24 tag 12345
     fabric forwarding mode anycast-gateway
```

In the above example, a gateway is created for each of the 2 tenant networks (Blue –L2 VNI 30000 and Red – L2 VNI 30001). End host traffic within a VNI (say 30000) is bridged, and traffic between tenant networks is routed. The routing takes place through a Layer-3 VNI (say 50000) typically having a one-on-one association with a VRF instance.

Forwarding between servers within a Layer-2 VNI

Figure 13: Packet Forwarding (Bridge)



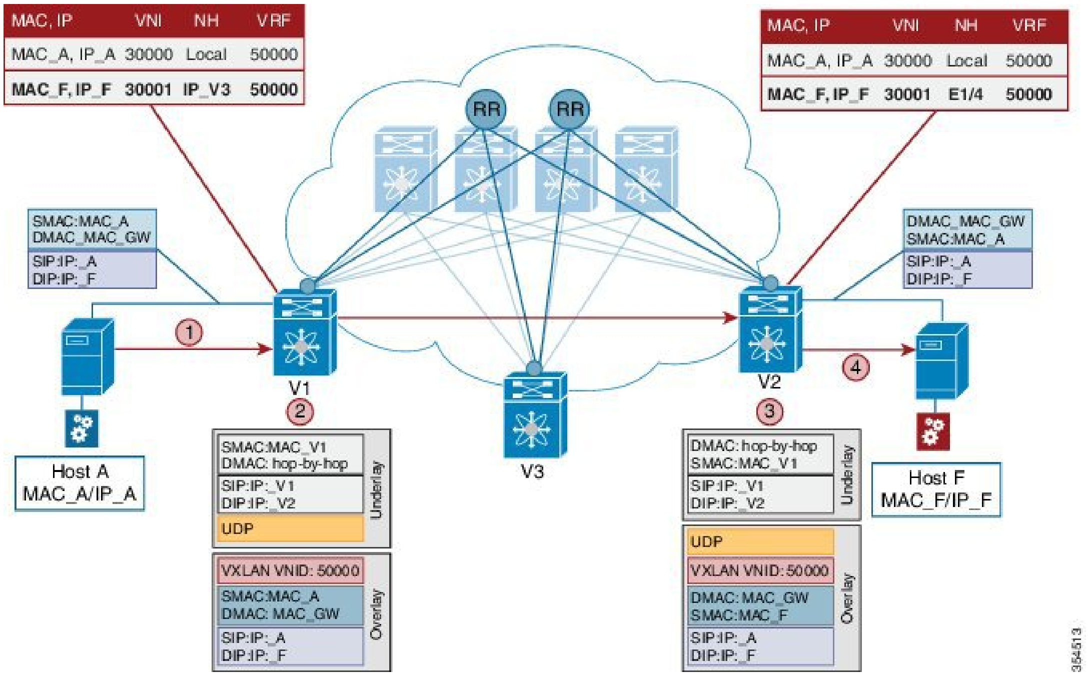
The VNI of the source end host, *Host A*, and the target end host, *Host B*, is 30000.

1. Host A sends traffic to the directly attached VTEP V1.
2. V1 performs a lookup based on the destination MAC address in the packet header (For communication that is bridged, the target end host's MAC address is updated in the DMAC field).
3. VTEP V1 bridges the packets and sends it toward VTEP V2 with a VXLAN header stamped with the Layer 2 VNI 30000.
4. VTEP V2 receives the packets, and post decapsulation, lookup, bridges them to Host B.

Packet forwarding between servers belonging to different Layer-2 VNIs

In the below example, the source and target end hosts (Host A and Host F) belong to different Layer-2 virtual networks (with VNIs 30000 and 30001). So, the traffic flow is between subnets, and hence routed. The VRF VNI 50000 is used to route the traffic

Figure 14: Packet Forwarding (Route)



A high level overview of the flow is given below:

1. Host A sends traffic to its default gateway (post ARP resolution) which is configured on the directly attached VTEP V1.
2. V1 performs a FIB lookup based on the destination IP address in the packet header.
3. VTEP V1 routes the packets and sends it toward VTEP V2 with a VXLAN header stamped with the VRF (Layer 3) VNI 50000.
4. VTEP V2 receives the packets, and post decapsulation, routing lookup, and rewrite, sends them to Host F.

Sample configurations for a setup with VNIs 30000 and 30001 are given below:

Configuration Example for VLAN, VNI, and VRF

VLAN to VNI mapping (MT-Lite)

(config) #

```
vlan 43
 vn-segment 30000

vlan 55
 vn-segment 30001
```

Allocate VLAN for VRF VNI

(config) #

```
vlan 500
 vn-segment 50000
```

VRF configuration for "customer" VRF

(config) #

```
vrf context VRF-A
   vni 50000
   rd auto
   address-family ipv4 unicast
      route-target both auto evpn
```

Configuration Example for MP-BGP EVPN

BGP Configuration for VRF (routing)

(config) #

```
router bgp 65535
  address-family ipv4 unicast
  neighbor RR_IP remote-as 65535
    address-family ipv4 unicast
    address-family l2vpn evpn
      send-community extended
```

(config) #

```
vrf VRF-A
  address-family ipv4 unicast
    advertise l2vpn evpn
```

EVPN Configuration for VNI (bridging)

(config) #

```
evpn
  vni 30000 l2
    rd auto
    route-target both auto
```

Type **Exit** twice.

(config-evpn) #

```
  vni 30001 l2
   rd auto
   route-target both auto
```

Routing at the VTEP - A high level view

Mandatory configurations

1. A VLAN is configured for each segment - sending segment, VRF segment and receiving segment.
2. BGP and EVPN configurations ensure redistribution of this information across the VXLAN setup.

Real time behavior

1. The source VTEP receives traffic and takes the routing decision. It then stamps the packet with the associated VRF VNI while sending traffic to the destination VTEP, which in turn forwards traffic to the destination server

Communication between a VXLAN overlay and an external network

The data center interconnect (DCI) functionality is implemented on the border device (leaf or spine) of the VXLAN EVPN network. Depending on the type of hand-off to the outside network such as MPLS, LISP, layer-2, and so on, appropriate DCI configuration is required on the border device(s) and the connecting edge device(s) of the outside network.

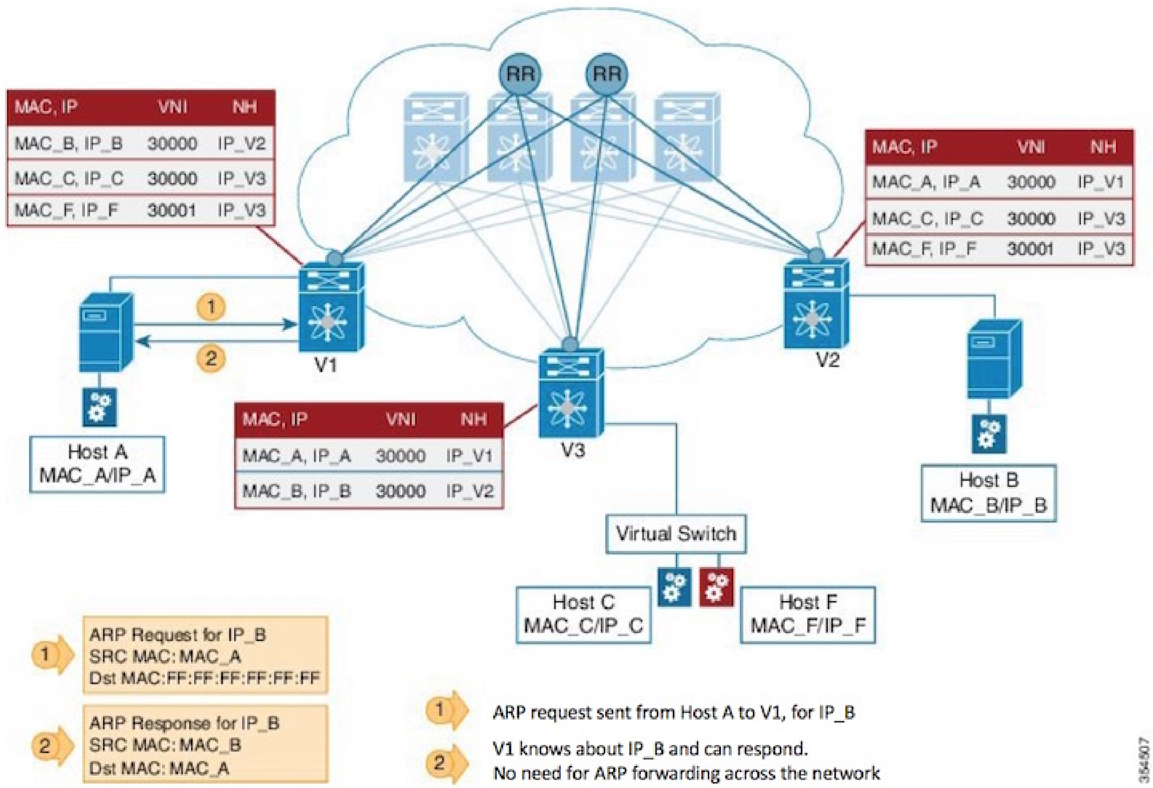
The DCI functionality is detailed in the *External Connectivity* chapters/use cases.

ARP Suppression

The following section illustrates ARP suppression functionality at VTEP V1 (Refer the *ARP Suppression* image, given below). ARP suppression is an enhanced function configured under the layer-2 VNI (using the **suppress-arp** command). Essentially, the IP-MACs learnt locally via ARP as well as those learnt over BGP-EVPN are stored in a local ARP suppression cache at each ToR. ARP request sent from the end host is trapped at the source ToR. A lookup is performed in the ARP suppression cache with the destination IP as the key. If there is a HIT, then the ToR proxies on behalf of the destination with the destination MAC. This is the case depicted in the below image.

In case the lookup results in a MISS, when the destination is unknown or a silent end host, the ToR re-injects the ARP request received from the requesting end host and broadcasts it within the layer-2 VNI. This entails sending the ARP request out locally over the server facing ports as well as sending a VXLAN encapsulated packet with the layer-2 VNI over the IP core. The VXLAN encapsulated packet will be decapsulated by every receiving VTEP that has membership within the same layer-2 VNI. These receiving VTEPs will then forward the inner ARP frame toward the server facing ports. Assuming that the destination is alive, the ARP request will reach the destination, which in turn will send out an ARP response toward the sender. The ARP response is trapped by the receiving ToR, even though ARP response is a unicast packet directed to the source VM, since the ARP-suppression feature is enabled. The ToR will learn about the destination IP/MAC and in turn advertise it over BGP-EVPN to all the other ToRs. In addition, the ToR will reinject the ARP response packet into the network (VXLAN-encapsulate it toward the IP core since original requestor was remote) so that it will reach the original requestor.

Figure 15: ARP Suppression



MAC, IP	VNI	NH
MAC_B, IP_B	30000	IP_V2
MAC_C, IP_C	30000	IP_V3
MAC_F, IP_F	30001	IP_V3

MAC, IP	VNI	NH
MAC_A, IP_A	30000	IP_V1
MAC_C, IP_C	30000	IP_V3
MAC_F, IP_F	30001	IP_V3

MAC, IP	VNI	NH
MAC_A, IP_A	30000	IP_V1
MAC_B, IP_B	30000	IP_V2

1 ARP Request for IP_B
SRC MAC: MAC_A
Dst MAC:FF:FF:FF:FF:FF:FF

2 ARP Response for IP_B
SRC MAC: MAC_B
Dst MAC: MAC_A

1 ARP request sent from Host A to V1, for IP_B

2 V1 knows about IP_B and can respond.
No need for ARP forwarding across the network

Unknown unicast (packet) suppression

Typically, an unknown unicast scenario arises when an end host has resolved the ARP but the MAC address of the end host is not available/updated in the switch.

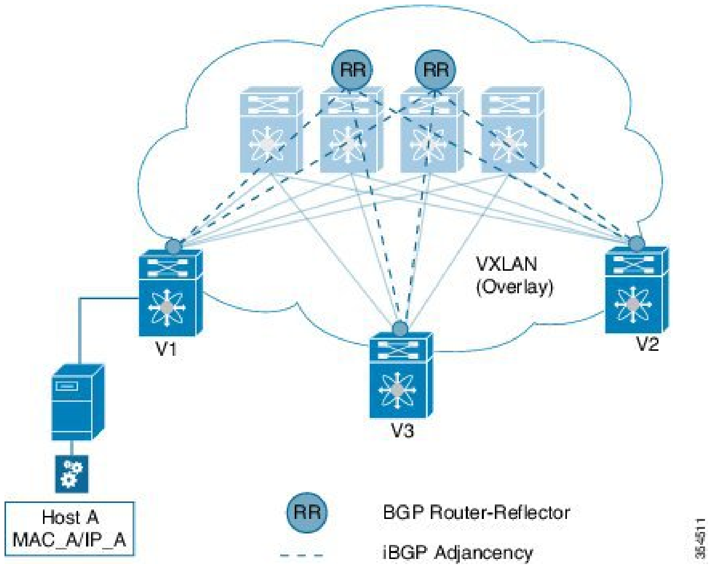
Unknown unicast traffic from an end host is by default flooded in the VLAN. It is possible to avoid the flooding of this traffic to the overlay network without affecting the flooding of this traffic on local host/sever ports attached to the ToR switch. Use the **suppress-unknown-unicast** command to do the same.

The *suppress unknown unicast* function is supported on ToRs/VTEPs in a VXLAN EVPN fabric. This function allows flooding of traffic within the attached switch by including local host/server ports attached to the ToR switch in the output interface index flood list (OIFL) and excluding overlay Layer-3 ports in the hardware.

Performing End Host Detection, Deletion and Move

End host detection by a VTEP device

Figure 16: End Host Detection



When a new end host (*Host A*) is attached to VTEP V1, the following actions occur:

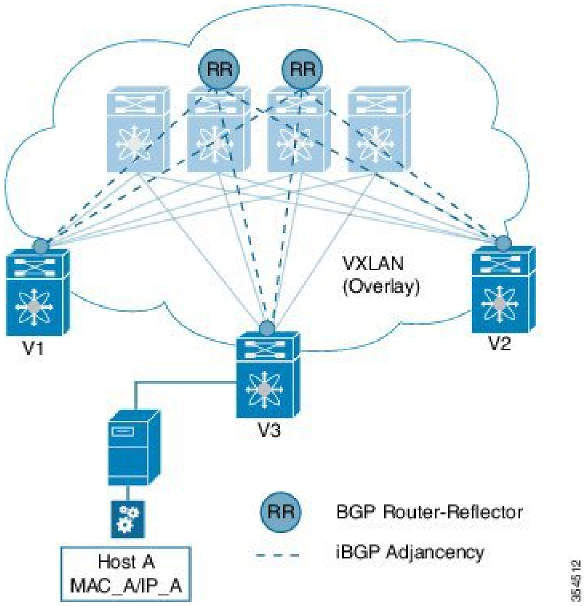
1. VTEP V1 learns Host A's MAC and IP address (MAC_A and IP_A).
2. V1 advertises MAC_A and IP_A to the other VTEPs V2 and V3 through the route reflector.
3. The choice of encapsulation (VXLAN) is also advertised.

A sample depiction of Host A related information is given below:

Table 1: Host A - Address Distribution Parameters

MAC, IP	L2VNI	L3VNI	Next-Hop	Encapsulation	Sequence
MAC_A, IP_A	30000	50000	IP_V1	VXLAN	0

Figure 17: Host move from V1 to V3



If Host A moves from VTEP V1 to V3, the following actions occur:

1. V3 detects Host A and advertises it with Sequence 1 (updating the previous instance of the sequence, *0*). The next hop IP address is reassigned to that of VTEP 3.

Table 2: Host A – Updated Parameters

MAC, IP	L2VNI	L3VNI	Next-Hop	Encapsulation	Sequence
MAC_A, IP_A	30000	50000	IP_V3	VXLAN	1

2. VTEP V1 detects a more recent route and withdraws its advertisement.

Mobility in a vPC scenario

In a vPC scenario where 2 ToR switches are vPC peers, whether the end host is attached to an orphan port or has a dual homed connection, the VIP address is advertised in the control plane and data plane, and the VIP address is carried in the (outer) source IP address field of the VXLAN packet.

Note VIP is a common, virtual VTEP IP address that is used for (unicast and multi destination) communication to and from the two switches in the vPC setup. The VIP address represents the two switches in the vPC setup, and is designated as the next hop address (for end hosts in the vPC domain) for reachability purposes.

Mobility across fabrics

To move end hosts (VMs) between VXLAN BGP EVPN fabrics, the fabrics should be connected through a DCI functionality. The various DCI functions are detailed in the *External Connectivity* chapters/use cases.

Multi-Destination Traffic

There are two options to transport tenant multi-destination traffic in the Programmable Fabric:

1. Through a shared multicast tree using PIM (ASM, SSM or BiDiR).
2. Through ingress replication (*Available for Cisco Nexus 9000 Series switches only*).

Refer to the table for Nexus switch type-to-BUM traffic support option mapping:

If you are using this Nexus switch:	*Use this option for BUM traffic:*
Cisco Nexus 9000 Series	Traffic ingress replication or PIM ASM/SSM/BiDir **Note** Beginning in Cisco NX-OS Release 9.2(1), PIM BiDir is supported.
Cisco Nexus 7000 and 7700 /F3 Series	PIM ASM/SSM or PIM BiDir
Cisco Nexus 5600 Series	PIM BiDir

CHAPTER 3

Forwarding Configurations

Forwarding Configurations

Forwarding Configurations for Cisco Nexus 5600, 7000 and 9000 Series Switches in the Programmable Fabric

Use these configurations for configuring your Cisco Nexus 5600, 7000 and 9000 Series switches.

Note For ease of use, the configuration mode from which you need to start configuring a task is mentioned at the beginning of each configuration.

Cisco Nexus 5600 Series switch configuration

The following configurations are required for the Cisco Nexus 5600 Series switch for supporting BGP-EVPN with VXLAN overlay. Note that most of the configurations required for enabling VXLAN remain the same, EVPN configurations are what will be the emphasis here:

1. Initial configuration - Install the network virtualization overlay, BGP, and EVPN features on the VTEPs.
2. Implement Layer 2 VNI configurations for tenant networks within a tenant.
3. Implement Layer 3 VNI configurations for the tenant.

Note Though configuration examples are mainly IPv4, IPv6 addresses are also supported in the VXLAN EVPN fabric.

Note A Cisco Nexus 5600 Series switch enabled for switching-mode store-and-forward may experience a egress ASIC buffer stuck under the following conditions:

- An unsolicited **write erase** command is issued. Additional parameters are configured that have impact on the forwarding decision, which means a new VLAN or VNI is created.
- A **copy running-configuration startup configuration** command is issued after the extra parameters are configured, and then additional new VLAN or VNI parameters are configured again.
- In the reported instance of this problem the switch was configured as a vPC peer switch and the issue affected the ASIC that holds the connection to the vPC peer-link.

To avoid this problem do not use random **write-erase** commands. If such a command was issued in error, immediately run the **copy running-configuration startup-configuration** command.

Initial configuration

(config) #

```
install feature-set fabric
feature-set fabric
feature fabric forwarding
feature interface-vlan
feature ospf
OR
feature isis
```

Attention You can use either OSPF or IS-IS as the routing protocol.

(config) #

```
feature nv overlay
feature bgp
feature vn-segment-vlan-based
nv overlay evpn
```

Configure the anycast gateway MAC address

(config) #

```
fabric forwarding anycast-gateway-mac 2020.0000.00aa
```

Configure BGP L2VPN EVPN address family

(config) #

```
router bgp 100
  neighbor 10.1.1.53 remote-as 100
```

```
    update-source loopback0
    address-family l2vpn evpn
      send-community both
```

Layer 2 VNI configurations for a tenant network

Associate a VLAN to the Layer 2 VNI

(config) #

```
vlan 200
  vn-segment 30000
```

Create a loopback interface for BGP and assign an IP address to it

(config) #

```
interface loopback 0
   ip address 10.1.1.54/32
```

Create a loopback interface for NVE and assign an IP address to it

(config) #

```
interface loopback 1
   ip address 10.1.2.54/32
```

Associate the Layer 2 VNI to the overlay and configure multicast group membership

(config) #

```
interface nve 1
 no shutdown
 source-interface loopback1
 host-reachability protocol bgp
 member vni 30000
   suppress-arp
   mcast-group 239.1.1.0
```

Associate the Layer 2 VNI to the EVPN address family, and enable route distinguisher and route target functions for the VNI

(config) #

```
evpn
  vni 30000 l2
    rd auto
    route-target import auto
    route-target export auto
```

Note Alternatively, the following config can also be used:

```
evpn
  vni 30000 l2
    rd auto
    route-target both auto
```

The combination of the **router BGP** command (configured earlier) and the **evpn** command ensures that BGP EVPN is configured to advertise 'MAC route' or 'MAC + associated host routes' of servers attached to the VTEP, for the specified Layer 2 VNI (Route type 2 [Refer to the EVPN RFC document for more details]). By default, the MAC route will be advertised, and the associated host route will be advertised if either there is an SVI configured for that VLAN in anycast-gateway mode or if suppress-arp option is enabled for that L2 VNI (See *ARP Suppression* section).

In the above NVE example, the MAC+IP routes for the hosts are advertised into BGP-EVPN for hosts belonging to layer 2 VNI 30000.

Layer 3 VNI configurations for a tenant

Associate the VRF VNI (Layer 3 VNI) to the customer VRF.

Enable VRF route distinguisher and route target functions.

(config) #

```
vrf context coke
  vni 50000
  rd auto
  address-family ipv4 unicast
     route-target both auto evpn
```

In the above example, the option *both* is used to import and export routes associated with the Layer 3 VNI 50000.

Associate the VRF VNI to a VLAN and associate an SVI to the customer VRF

(config) #

```
vlan 2200
   vn-segment 50000
```

(config) #

```
interface vlan 2200
   vrf member coke
   ip forward
   ipv6 forward
   no ip redirects
   no ipv6 redirects
   no shutdown
```

In order to avoid the overhead of creating a core facing vlan and corresponding SVI on a per vrf basis, we also provide an option of using a vrf-tenant-profile that automatically takes care of this. Note that if there is a **vrf-tenant-profile** configured, then the user must ensure the following CLIs related to dynamic and core-VLANs are also enabled.

(config) #

```
system fabric dynamic-vlans 100-2400
system fabric core-vlans 100-300
```

switch #

```
configure profile vrf-tenant-profile
  vlan $vrfVlanId
    vn-segment $vrfSegmentId
  interface vlan $vrfVlanId
    vrf member $vrfName
      ip forward
      ipv6 forward
      no ip redirects
      no ipv6 redirects
      no shutdown
end
```

Add the Layer 3 VRF VNI to the overlay network

(config) #

```
interface nve 1
  host-reachability protocol bgp
  member vni 50000 associate-vrf
```

Associate the customer VRF to BGP and enable L2VPN EVPN route distribution

(config) #

```
router bgp 100
  vrf coke
    address-family ipv4 unicast
      advertise l2vpn evpn
```

Enable host/server facing SVI (and associate it to a VRF) for Layer 3 connectivity on the distributed anycast gateway

(config) #

```
interface vlan 200
  vrf member coke
  ip address 209.165.202.129/27
  fabric forwarding mode anycast-gateway
```

Cisco Nexus 5600 Series switches verification

For verification of MAC routes, refer these commands:

```
switch# show mac address-table dynamic

Legend:
        * - primary entry, G - Gateway MAC, (R) - Routed MAC, O - Overlay MAC
        age - seconds since last seen,+ - primary entry using vPC Peer-Link
```

```
   VLAN     MAC Address      Type      age     Secure NTFY   Ports/SWID.SSID.LID
---------+-----------------+--------+---------+------+----+------------------
* 200      2010.0000.0010    dynamic   270      F      F     Eth100/1/1
* 200      2010.0000.0011    dynamic   0        F      F     nve1/10.1.1.56
* 200      2010.0000.0012    dynamic   0        F      F     nve1/10.1.1.74
* 200      2010.0000.0013    dynamic   0        F      F     nve1/10.1.1.56
* 200      8080.c800.0038    dynamic   0        F      F     nve1/10.1.1.74
* 1        24e9.b392.316b    dynamic   1190     F      F     Eth100/1/1
```

```
switch# show l2route evpn mac all

Topology    Mac Address    Prod   Next Hop (s)
----------- -------------- ------ ---------------
200         2010.0000.0010 Local  Eth100/1/1
200         2010.0000.0011 BGP    10.1.1.56
200         2010.0000.0012 BGP    10.1.1.74
200         2010.0000.0013 BGP    10.1.1.56
200         8080.c800.0038 BGP    10.1.1.74
2200        002a.6ab2.0181 VXLAN  10.1.1.56
2200        8c60.4f14.2efc VXLAN  10.1.1.74
```

Command output description

Prod (producer) column displays the source of origination of the MAC address.

Local means a MAC address learnt locally via a server facing or edge port, **BGP** means the remote end host MAC was learnt from a remote VTEP via BGP-EVPN and **VXLAN** indicates the router MAC of the remote VTEP as carried in the extended community in the BGP advertisement.

```
switch# show bgp l2vpn evpn

BGP routing table information for VRF default, address family L2VPN EVPN
BGP table version is 198, local router ID is 10.1.1.54
Status: s-suppressed, x-deleted, S-stale, d-dampened, h-history, *-valid, >-best
Path type: i-internal, e-external, c-confed, l-local, a-aggregate, r-redist, I-injected
Origin codes: i - IGP, e - EGP, ? - incomplete, | - multipath, & - backup

   Network            Next Hop            Metric     LocPrf     Weight Path
Route Distinguisher: 10.1.1.54:32967    (L2VNI 30000)
*>l[2]:[0]:[0]:[48]:[2010.0000.0010]:[0]:[0.0.0.0]/216
                      10.1.1.54                          100      32768 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[0]:[0.0.0.0]/216
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[0]:[0.0.0.0]/216
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[8080.c800.0038]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
*>l[2]:[0]:[0]:[48]:[2010.0000.0010]:[32]:[200.0.0.10]/272
                      10.1.1.54                          100      32768 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[200.0.0.11]/272
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[200.0.0.12]/272
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[200.0.0.13]/272
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[8080.c800.0038]:[32]:[200.0.0.56]/272
                      10.1.1.74                          100          0 i

Route Distinguisher: 10.1.1.56:3
```

```
*>i[5]:[0]:[0]:[24]:[209.165.202.130]:[0.0.0.0]/224
                      10.1.1.56                 0        100          0 ?

Route Distinguisher: 10.1.1.56:32967
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[0]:[0.0.0.0]/216
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[0]:[0.0.0.0]/216
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[209.165.202.140]/272
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[209.165.202.142]/272
                      10.1.1.56                          100          0 i
Route Distinguisher: 10.1.1.74:32967
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[8080.c800.0038]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[209.165.202.141]/272
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[8080.c800.0038]:[32]:[209.165.202.143]/272
                      10.1.1.74                          100          0 i
```

```
switch# show nve peers

Interface Peer-IP    State LearnType Uptime   Router-Mac
--------- ---------------  ----- --------- -------- -----------------
nve1      10.1.1.56   Up    CP        1d12h   002a.6ab2.0181
nve1      10.1.1.74   Up    CP        1d12h   8c60.4f14.2efc
```

For verification of IP host and prefix routes, refer these commands:

```
switch# show ip arp vrf coke

Flags: * - Adjacencies learnt on non-active FHRP router
       + - Adjacencies synced via CFSoE
       # - Adjacencies Throttled for Glean
       D - Static Adjacencies attached to down interface

IP ARP Table for context coke
Total number of entries: 1
Address         Age       MAC Address      Interface
209.165.202.13  00:18:23  2010.0000.0010   Vlan200
```

```
switch# show ip route vrf coke

IP Route Table for VRF "coke"
'*' 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
    *via 10.1.1.1, Vlan10, [0/0], 1d12h, direct
10.1.1.1/32, ubest/mbest: 1/0, attached
    *via 10.1.1.1, Vlan10, [0/0], 1d12h, local
209.165.202.130/27, ubest/mbest: 1/0, attached
    *via 209.165.202.129, Vlan200, [0/0], 1d12h, direct, tag 12345,
209.165.202.129/32, ubest/mbest: 1/0, attached
    *via 209.165.202.129, Vlan200, [0/0], 1d12h, local, tag 12345,
209.165.202.139/32, ubest/mbest: 1/0, attached
    *via 209.165.202.139, Vlan200, [190/0], 1d12h, hmm
209.165.202.140 /32, ubest/mbest: 1/0
    *via 10.1.1.56%default, [200/0], 1d12h, bgp-100, internal, tag 100,  (mpls-vpn)segid
50000 tunnel: 16843064 encap: 1
```

Command output description

Direct means that the subnet prefix is configured locally under a Layer-3 interface on this switch. **Local** means the IP address belongs to the switch aka locally configured under a Layer-3 interface on that switch (10.1.1.254/24).

```
switch# show l2route evpn mac-ip all

Topology ID Mac Address    Prod     Host IP           Next Hop(s)
----------- -------------- ---- --------------------------------------- --------
-------
200         2010.0000.0010 HMM      209.165.202.139   N/A

200         2010.0000.0011 BGP      209.165.202.140   10.1.1.56

200         2010.0000.0012 BGP      209.165.202.141   10.1.1.74

200         2010.0000.0013 BGP      209.165.202.142   10.1.1.56

200         8080.c800.0038 BGP      209.165.202.143   10.1.1.74
```

```
switch# show bgp l2vpn evpn

Route Distinguisher: 10.1.1.54:3    (L3VNI 50000)
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[209.165.202.144]/272
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[209.165.202.141]/272
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[209.165.202.143]/272
                      10.1.1.56                          100          0 i
*>l[5]:[0]:[0]:[24]:[209.165.202.130]:[0.0.0.0]/224
                      10.1.1.54                0        100      32768 ?
* i                   10.1.1.56                0        100          0 ?
```

Cisco Nexus 7000 Series switch configuration

The following BGP, EVPN and overlay configurations are required for the Cisco Nexus 7000 Series and 7700 Series switches with F3 or M3 cards:

1. Initial configuration - Install the network virtualization overlay, BGP, and EVPN features on the VTEPs.
2. Layer 2 VNI configurations for tenant networks within a tenant.
3. Layer 3 VNI configurations for a tenant.

Note Though configuration examples are mainly IPv4, IPv6 addresses are also supported in the VXLAN EVPN fabric.

VXLAN BGP EVPN configuration for the Cisco Nexus 7000 Series switches is also available here. While the 7.2 release only supported the border leaf and border spine functionality, the 7.3 version in addition also supports the leaf functionality.

A switch VDC with M3 modules cannot perform the role of a VXLAN BGP EVPN leaf switch.

Initial configurations

(config) #

```
install feature-set fabric
feature-set fabric
feature fabric forwarding
feature interface-vlan
feature ospf
OR
feature isis
```

Attention You can use either OSPF or IS-IS as the underlay routing protocol.

Note The **install feature-set fabric** command should only be used in the admin VDC. When using a VDC, ensure the VDC is of type F3 or M3, for EVPN. A sample configuration is given below:

(config) #

```
vdc test
    limit-resource module-type f3
```

(config) #

```
feature nv overlay
feature bgp
feature vni
nv overlay evpn
```

Configure the anycast gateway MAC address

(config) #

```
fabric forwarding anycast-gateway-mac 2020.0000.00aa
```

Configure BGP L2VPN EVPN address family

(config) #

```
router bgp 100
   neighbor 10.1.1.53 remote-as 100
     update-source loopback0
     address-family l2vpn evpn
        send-community both
```

Layer 2 VNI configurations for a tenant network

Create a bridge domain and associate the Layer 2 VNI with it

(config) #

```
vni 30000
system bridge-domain 200-210
bridge-domain 200
   member vni 30000
```

While the **system bridge-domain** command identifies the bridge domain IDs, the **bridge-domain** command configures the specified bridge domain(s).

Associate a VLAN (or dot1q tag) with the Layer 2 VNI:

(config) #

```
encapsulation profile vni cisco
  dot1q 50 vni 30000
```

Note For an access port, you should use the **untagged** keyword, as shown below.

```
encapsulation profile vni ACCESS
  untagged vni 30000
```

Associate the encapsulation profile with the server facing interface

(config) #

```
interface Ethernet 1/12
   no shutdown
   no switchport
   service instance 1 vni
   encapsulation profile cisco default
      no shutdown
```

Create a loopback interface for BGP and assign an IP address to it

(config) #

```
interface loopback 0
   ip address 10.1.1.54/32
```

Create a loopback interface for NVE and assign an IP address to it

(config) #

```
interface loopback 1
```

```
   ip address 10.1.2.54/32
```

Associate the Layer 2 VNI to the overlay and configure multicast group membership

(config) #

```
interface nve 1
   no shutdown
   source-interface loopback0
   host-reachability protocol bgp
   member vni 30000
      suppress-arp
      mcast-group 239.1.1.0
```

Enable EVPN and associate the Layer 2 VNI to it

Enable route distinguisher and route target functions for the Layer 2 VNI

(config) #

```
evpn
  vni 30000 l2
    rd auto
    route-target import auto
    route-target export auto
```

Note that with the Cisco Nexus 7000 Series switches, a VNI is associated with a bridge-domain (1:1). Refer to the respective configuration guide for more information on bridge-domains. The combination of the **router BGP** command (configured earlier) and the **evpn** command ensures that BGP EVPN is configured to advertise ‘MAC route’ or ‘MAC + associated host routes’ of servers attached to the VTEP, for the specified Layer 2 VNI.

In the above NVE example, MAC+ IP routes are advertised into BGP-EVPN for hosts belonging to layer 2 VNI 30000.

Layer 3 VNI configurations for a tenant

Associate the VRF VNI to the customer VRF

Enable VRF route distinguisher and VRF route target functions for the Layer 3 VNI

(config) #

```
vrf context coke
   vni 50000
   rd auto
   address-family ipv4 unicast
      route-target both auto evpn
```

In the above example, the option *both* is used to import and export routes associated with the Layer 3 VNI 50000. Specifically, the layer-3 routes will be advertised with route-target 100:50000 where 100 is the BGP Autonomous system number and 50000 is the layer-3 VNI.

Associate the VRF VNI to a bridge-domain and associate a BDI to the customer VRF

(config) #

```
system bridge-domain add 2200
vni 50000
bridge-domain 2200
  member vni 50000

interface bdi2200
  vrf member coke
  ip forward
  no ip redirects
  no shutdown
```

While the **system bridge-domain** command identifies the bridge domain IDs, the **bridge-domain** command configures the specified bridge domain(s).

Add the Layer 3 VRF VNI to the overlay network and enable BGP reachability

(config) #

```
interface nve 1
   host-reachability protocol bgp
   member vni 50000 associate-vrf
```

Configure BGP, associate the customer VRF to BGP and enable L2VPN EVPN route distribution

(config) #

```
router bgp 100
   vrf coke
      address-family ipv4 unicast
         advertise l2vpn evpn
```

Enable host/server facing BDI (and associate it to a VRF) for Layer 3 connectivity on the distributed anycast gateway

(config) #

```
interface bdi200
   vrf member coke
   ip address 10.1.1.1/24
   fabric forwarding mode anycast-gateway
   no shutdown
```

Cisco Nexus 7000 Series switches verification

For verification of MAC routes, refer these commands:

The following is sample output to verify that end host MAC addresses (local and remote) are added to the MAC address table:

```
switch# show mac address-table dynamic

Note: MAC table entries displayed are getting read from software.
 Use the 'hardware-age' keyword to get information related to 'Age'

 Legend:
        * - primary entry, G - Gateway MAC, (R) - Routed MAC, O - Overlay MAC
        age - seconds since last seen,+ - primary entry using vPC Peer-Link, E -
 EVPN entry
        (T) - True, (F) - False ,  ~~~ - use 'hardware-age' keyword to retrieve
age info

VLAN/BD   MAC Address      Type      age     Secure NTFY Ports/SWID.SSID.LID
---------+-----------------+--------+---------+------+----+------------------

* 200      2010.0000.0010    dynamic   270      F    F   Eth100/1/1
* 200      2010.0000.0011    dynamic   0        F    F   nve1/10.1.1.56
* 200      2010.0000.0012    dynamic   0        F    F   nve1/10.1.1.74
* 200      2010.0000.0013    dynamic   0        F    F   nve1/10.1.1.56
* 200      8080.c800.0038    dynamic   0        F    F   nve1/10.1.1.74
* 1        24e9.b392.316b    dynamic   1190     F    F   Eth100/1/1
```

The following is sample output for viewing MAC addresses of end hosts across all EVPN instances (EVIs) pertaining to the switch:

```
switch# show l2route evpn mac all

Topology    Mac Address    Prod   Next Hop (s)
----------- -------------- ------ ---------------
200         2010.0000.0010 Local  Eth100/1/1
200         2010.0000.0011 BGP    10.1.1.56
200         2010.0000.0012 BGP    10.1.1.74
200         2010.0000.0013 BGP    10.1.1.56
200         8080.c800.0038 BGP    10.1.1.74
2200        002a.6ab2.0181 VXLAN  10.1.1.56
2200        8c60.4f14.2efc VXLAN  10.1.1.74
```

The following sample output displays BGP routing table information for the L2VPN EVPN address family. It includes route distinguisher and next hop information.

```
switch # show bgp l2vpn evpn

BGP routing table information for VRF default, address family L2VPN EVPN
BGP table version is 198, local router ID is 10.1.1.54
Status: s-suppressed, x-deleted, S-stale, d-dampened, h-history, *-valid, >-best
Path type: i-internal, e-external, c-confed, l-local, a-aggregate, r-redist, I-injected
Origin codes: i - IGP, e - EGP, ? - incomplete, | - multipath, & - backup

   Network            Next Hop            Metric     LocPrf     Weight Path

Route Distinguisher: 10.1.1.54:32967    (L2VNI 30000)
*>l[2]:[0]:[0]:[48]:[2010.0000.0010]:[0]:[0.0.0.0]/216
                      10.1.1.54                          100      32768 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[0]:[0.0.0.0]/216
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[0]:[0.0.0.0]/216
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[8080.c800.0038]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
*>l[2]:[0]:[0]:[48]:[2010.0000.0010]:[32]:[209.165.202.139]/272
                      10.1.1.54                          100      32768 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[209.165.202.140]/272
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[209.165.202.141]/272
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[209.165.202.142]/272
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[8080.c800.0038]:[32]:[209.165.202.143]/272
```

```
                      10.1.1.74                          100          0 i

Route Distinguisher: 10.1.1.56:3
*>i[5]:[0]:[0]:[24]:[209.165.202.130]:[0.0.0.0]/224
                      10.1.1.56                 0        100          0 ?

Route Distinguisher: 10.1.1.56:32967
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[0]:[0.0.0.0]/216
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[0]:[0.0.0.0]/216
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[209.165.202.140]/272
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[209.165.202.142]/272
                      10.1.1.56                          100          0 i

Route Distinguisher: 10.1.1.74:32967
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[8080.c800.0038]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[209.165.202.141]/272
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[8080.c800.0038]:[32]:[209.165.202.143]/272
                      10.1.1.74                          100          0 i
```

The following sample output displays peer VTEP device information.

```
switch # show nve peers

Interface Peer-IP    State LearnType Uptime   Router-Mac
--------- ---------------  ----- --------- -------- -----------------
nve1      10.1.1.56         Up    CP        1d12h    002a.6ab2.0181
nve1      10.1.1.74         Up    CP        1d12h    8c60.4f14.2efc
```

For IP host and prefix routes verification, refer these commands:

The following sample output displays tenant (VRF) information

```
switch # show ip arp vrf coke

Flags: * - Adjacencies learnt on non-active FHRP router
       + - Adjacencies synced via CFSoE
       # - Adjacencies Throttled for Glean
       D - Static Adjacencies attached to down interface

IP ARP Table for context coke
Total number of entries: 1
Address          Age            MAC Address       Interface
209.165.202.144  00:18:23       2010.0000.0010    Bdi200
```

The following sample output displays tenant (VRF) information

```
switch # show ip route vrf coke

IP Route Table for VRF "coke"
'*' 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
    *via 10.1.1.1, Bdi10, [0/0], 1d12h, direct
```

```
10.1.1.1/32, ubest/mbest: 1/0, attached
    *via 10.1.1.1, Bdi10, [0/0], 1d12h, local
209.165.202.130/27, ubest/mbest: 1/0, attached
    *via 209.165.202.129, Bdi200, [0/0], 1d12h, direct, tag 12345,
209.165.202.129/32, ubest/mbest: 1/0, attached
    *via 209.165.202.129, Bdi200, [0/0], 1d12h, local, tag 12345,
209.165.202.139/32, ubest/mbest: 1/0, attached
    *via 209.165.202.139, Bdi200, [190/0], 1d12h, hmm
209.165.202.140 /32, ubest/mbest: 1/0
    *via 10.1.1.56%default, [200/0], 1d12h, bgp-100, internal, tag 100,  (mpls-vpn)segid
50000 tunnel: 16843064 encap: 1
```

The following sample output displays MAC - IP address binding for all attached and remote end hosts (learned through the BGP EVPN control plane).

```
switch # show l2route evpn mac-ip all

Topology ID Mac Address    Prod     Host IP           Next Hop(s)
----------- -------------- ---- --------------------------------------- --------
200         2010.0000.0010 HMM      209.165.202.139    N/A

200         2010.0000.0011 BGP      209.165.202.140    10.1.1.56

200         2010.0000.0012 BGP      209.165.202.141    10.1.1.74

200         2010.0000.0013 BGP      209.165.202.142    10.1.1.56

200         8080.c800.0038 BGP      209.165.202.143    10.1.1.74
```

The following sample output displays BGP routing table information for Layer-3 VNIs.

```
switch # show bgp l2vpn evpn

Route Distinguisher: 10.1.1.54:3    (L3VNI 50000)
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[209.165.202.144]/272
                      10.1.1.56                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[209.165.202.141]/272
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[209.165.202.143]/272
                      10.1.1.56                          100          0 i
*>l[5]:[0]:[0]:[24]:[209.165.202.130]:[0.0.0.0]/224
                      10.1.1.54                 0        100      32768 ?
* i                   10.1.1.56                 0        100          0 ?
```

Cisco Nexus 9000 Series switch configuration

The following configurations are required for a Cisco Nexus 9000 Series switch for the VXLAN BGP EVPN fabric.

1. Initial configuration - Install the network virtualization overlay, BGP, and EVPN features on the VTEPs.
2. Layer 2 VNI configurations for tenant networks within a tenant.
3. Layer 3 VNI configurations for a tenant.

Note Though configuration examples are mainly IPv4, IPv6 addresses are also supported in the VXLAN BGP EVPN fabric.

Initial configuration

(config) #

```
nv overlay evpn
feature bgp
feature ospf
OR
feature isis
```

Attention You can use either OSPF or IS-IS as the underlay routing protocol.

(config) #

```
feature interface-vlan
feature vn-segment-vlan-based
feature nv overlay
```

Configure the anycast gateway MAC address

(config) #

```
fabric forwarding anycast-gateway-mac 2020.0000.00aa
```

Configure BGP L2VPN EVPN address family

(config) #

```
router bgp 100
  neighbor 192.0.2.1
  remote-as 100
  update-source loopback0
  address-family l2vpn evpn
    send-community
    send-community extended
```

Layer 2 VNI configurations for a tenant network

Associate a VLAN to the Layer 2 VNI

(config) #

```
vlan 200
  vn-segment 30000
```

Create a loopback interface for BGP and assign an IP address to it

(config) #

```
interface loopback 0
```

```
   ip address 192.0.2.10/32
```

Create a loopback interface for NVE and assign an IP address to it

(config) #

```
interface loopback 1
   ip address 198.51.100.1/32
```

Associate the Layer 2 VNI to the overlay and configure multicast group membership

(config) #

```
interface nve 1
  no shutdown
  source-interface loopback1
  host-reachability protocol bgp
  member vni 30000
    suppress-arp
    mcast-group 239.1.1.0
```

Associate the Layer 2 VNI to the overlay and configure multicast group membership

(config)#

```
interface nve 1
  source-interface loopback 1
  host-reachability protocol bgp
  global suppress-arp
  global mcast-group 224.1.1.1 L2
  global mcast-gropu 225.1.1.1 L3
  member vni 10000
    suppress-arp disable
  member 20000 associate-vrf
    mcast-group 225.1.1.10
```

Associate the Layer 2 VNI to the EVPN address family, and enable route distinguisher and route target functions for the VNI

(config) #

```
evpn
  vni 30000 l2
    rd auto
    route-target import auto
    route-target export auto
```

Note Alternatively, the following configurations can also be used:

```
evpn
  vni 30000 l2
    rd auto
    route-target both auto
```

The combination of the **router BGP** command (configured earlier) and the **evpn** command ensures that BGP EVPN is configured to advertise 'MAC route' or 'MAC + associated host routes' of servers attached to the VTEP, for the specified Layer 2 VNI. (Route type 2 [Refer to the EVPN RFC document for more details]). By default, the MAC route will be advertised, and the associated host route will be advertised if there is an SVI configured for that VLAN in the anycast-gateway mode or if suppress-arp option is enabled for that L2 VNI (see *ARP Suppression* section).

In the above NVE example, MAC and IP routes are advertised into BGP-EVPN for end hosts belonging to layer 2 VNI 30000.

Layer 3 VNI configurations for a tenant

Associate the VRF VNI (Layer 3 VNI) to the customer VRF

Enable VRF route distinguisher and route target functions

(config) #

```
vrf context coke:vrf1
  vni 50000
  rd auto
  address-family ipv4 unicast
    route-target both auto
    route-target both auto evpn
```

In the above example, the option *both* is used to import and export routes associated with the Layer 3 VNI 50000.

Associate the VRF VNI to a VLAN and associate an SVI to the customer VRF

(config) #

```
vlan 2500
  vn-segment 50000
```

(config) #

```
interface vlan 2500
  vrf member coke:vrf1
  ip forward
  ipv6 forward
  no ip redirects
  no ipv6 redirects
  no shutdown
```

In order to avoid the overhead of creating a core facing VLAN and corresponding SVI on a per VRF basis, the vrf-tenant-profile (that automatically takes care of this) is provided. If you configure a **vrf-tenant-profile**, you should enable the following CLIs related to dynamic and core VLANs.

(config) #

```
system fabric dynamic-vlans 2500-3500
system fabric core-vlans 2500-2999
```

```
configure profile vrf-tenant-profile
   vlan $vrfVlanId
     vn-segment $vrfSegmentId
   interface vlan $vrfVlanId
     vrf member $vrfName
     ip forward
     ipv6 forward
     no ip redirects
     no ipv6 redirects
     no shutdown
end
```

Add the Layer 3 VRF VNI to the overlay network

(config) #

```
interface nve 1
   host-reachability protocol bgp
   member vni 50000 associate-vrf
```

Associate the customer VRF to BGP and enable L2VPN EVPN route distribution

(config) #

```
router bgp 100
  vrf coke:vrf1
    address-family ipv4 unicast
      advertise l2vpn evpn
```

Enable host/server facing SVI (and associate it to a VRF) for Layer 3 connectivity on the distributed anycast gateway

(config) #

```
interface vlan 200
  vrf member coke:vrf1
  ip address 203.0.113.3/24 tag 12345
  fabric forwarding mode anycast-gateway
```

Cisco Nexus 9000 Series switches verification

For verification of MAC routes, refer these commands

The following is sample output to verify that end host MAC addresses (local and remote) are added to the MAC address table:

```
switch# show mac address-table dynamic

Legend:
        * - primary entry, G - Gateway MAC, (R) - Routed MAC, O - Overlay MAC
        age - seconds since last seen,+ - primary entry using vPC Peer-Link,
        (T) - True, (F) - False, C - ControlPlane MAC

   VLAN     MAC Address      Type      age     Secure NTFY Ports
---------+-----------------+--------+---------+------+----+------------------
*    1     a036.9f22.a277   dynamic  0         F      F    Eth1/7
```

```
C  200     002a.6a85.a67c   dynamic  0         F      F    nve1(198.51.100.10)
*  200     2010.0000.0012   dynamic  0         F      F    Eth1/7
C  200     2010.0000.0015   dynamic  0         F      F    nve1(198.51.100.10)
```

The following is sample output for viewing MAC addresses of end hosts across all EVPN instances (EVIs) pertaining to the switch:

```
switch# show l2route evpn mac all

Topology    Mac Address    Prod   Flags         Seq No     Next-Hops
----------- -------------- ------ ------------- ---------- ----------------
200         002a.6a85.a67c BGP    SplRcv        0          198.51.100.10
200         2010.0000.0012 Local  L,            0          Eth1/7
200         2010.0000.0015 BGP    SplRcv        0          198.51.100.10
2500        7c0e.ceca.f2ff VXLAN  Rmac          0          198.51.100.10
```

Command output description

Prod (producer) column displays the source of origination of the MAC address.

Local means a MAC address learnt locally via a server facing or edge port, BGP means the remote end host MAC was learnt from a remote VTEP via BGP-EVPN and VXLAN indicates the router MAC of the remote VTEP as carried in the extended community in the BGP advertisement.

The following sample output displays BGP routing table information for the L2VPN EVPN address family. It includes route distinguisher and next hop information:

```
switch # show bgp l2vpn evpn

BGP routing table information for VRF default, address family L2VPN EVPN
BGP table version is 26, local router ID is 192.0.2.10

Status: s-suppressed, x-deleted, S-stale, d-dampened, h-history, *-valid, >-best
Path type: i-internal, e-external, c-confed, l-local, a-aggregate, r-redist, I-i
njected
Origin codes: i - IGP, e - EGP, ? - incomplete, | - multipath, & - backup

   Network            Next Hop            Metric     LocPrf     Weight Path
Route Distinguisher: 192.0.2.20:3
*>i[5]:[0]:[0]:[24]:[203.0.113.6]:[0.0.0.0]/224
                     198.51.100.10         0        100          0 ?

Route Distinguisher: 192.0.2.20:32967
*>i[2]:[0]:[0]:[48]:[002a.6a85.a67c]:[0]:[0.0.0.0]/216
                     198.51.100.10                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0015]:[0]:[0.0.0.0]/216
                     198.51.100.10                          100          0 i
*>i[2]:[0]:[0]:[48]:[002a.6a85.a67c]:[32]:[200.0.0.52]/272
                     198.51.100.10                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0015]:[32]:[200.0.0.15]/272
                     198.51.100.10                          100          0 i

Route Distinguisher: 192.0.2.30:3
*>i[5]:[0]:[0]:[24]:[200.0.0.0]:[0.0.0.0]/224
                     198.51.100.10                 0        100          0 ?

Route Distinguisher: 192.0.2.30:32967
*>i[2]:[0]:[0]:[48]:[002a.6a85.a67c]:[0]:[0.0.0.0]/216
                     198.51.100.10                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0015]:[0]:[0.0.0.0]/216
                     198.51.100.10                          100          0 i
*>i[2]:[0]:[0]:[48]:[002a.6a85.a67c]:[32]:[200.0.0.52]/272
                     198.51.100.10                          100          0 i
```

```
*>i[2]:[0]:[0]:[48]:[2010.0000.0015]:[32]:[200.0.0.15]/272
                     198.51.100.10                          100          0 i

Route Distinguisher: 192.0.2.10:32967    (L2VNI 30000)
* i[2]:[0]:[0]:[48]:[002a.6a85.a67c]:[0]:[0.0.0.0]/216
                     198.51.100.10                          100          0 i
*>i                  198.51.100.10                          100          0 i
*>l[2]:[0]:[0]:[48]:[2010.0000.0012]:[0]:[0.0.0.0]/216
                     192.0.2.10                             100      32768 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0015]:[0]:[0.0.0.0]/216
                     198.51.100.10                          100          0 i
* i                  198.51.100.10                          100          0 i
*>i[2]:[0]:[0]:[48]:[002a.6a85.a67c]:[32]:[203.0.113.12]/272
                     198.51.100.10                          100          0 i
* i                  198.51.100.10                          100          0 i
*>l[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[203.0.113.5]/272
                     192.0.2.10                         100      32768 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0015]:[32]:[203.0.113.8]/272
                     198.51.100.10                          100          0 i
* i                  198.51.100.10                          100          0 i

Route Distinguisher: 192.0.2.10:3    (L3VNI 50000)
*>i[2]:[0]:[0]:[48]:[002a.6a85.a67c]:[32]:[203.0.113.12]/272
                     198.51.100.10                          100          0 i
* i                  198.51.100.10                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0015]:[32]:[203.0.113.8]/272
                     198.51.100.10                          100          0 i
* i                  198.51.100.10                          100          0 i
* i[5]:[0]:[0]:[24]:[200.0.0.0]:[0.0.0.0]/224
                     198.51.100.10                 0        100          0 ?
* i                  198.51.100.10                 0        100          0 ?
*>l                  198.51.100.1                  0        100      32768 ?
```

The following sample output displays peer VTEP device information:

```
switch # show nve peers

Interface Peer-IP          State LearnType Uptime   Router-Mac
--------- ---------------  ----- --------- -------- -----------------
nve1      198.51.100.10    Up    CP        3d00h    7c0e.ceca.f2ff
```

For verification of IP host and prefix routes, refer these commands

The following sample output displays tenant (VRF) information:

```
switch # show ip arp vrf coke:vrf1

Flags: * - Adjacencies learnt on non-active FHRP router
       + - Adjacencies synced via CFSoE
       # - Adjacencies Throttled for Glean
       CP - Added via L2RIB, Control plane Adjacencies
       PS - Added via L2RIB, Peer Sync
       RO - Dervied from L2RIB Peer Sync Entry
       D - Static Adjacencies attached to down interface

IP ARP Table for context coke:vrf1
Total number of entries: 1
Address         Age       MAC Address     Interface       Flags
203.0.113.5     00:12:46  2010.0000.0012  Vlan200
```

The following sample output displays tenant (VRF) information:

```
switch # show ip route vrf coke:vrf1

IP Route Table for VRF "coke:vrf1"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

203.0.113.6/24, ubest/mbest: 1/0, attached
    *via 203.0.113.3, Vlan200, [0/0], 3d00h, direct, tag 12345
203.0.113.3/32, ubest/mbest: 1/0, attached
    *via 203.0.113.3, Vlan200, [0/0], 3d00h, local, tag 12345
203.0.113.5/32, ubest/mbest: 1/0, attached
    *via 203.0.113.5, Vlan200, [190/0], 00:14:04, hmm
203.0.113.8/32, ubest/mbest: 1/0
    *via 198.51.100.10%default, [200/0], 3d00h, bgp-100, internal, tag 100 (evpn) segid:
50000 tunnelid: 0x16020202 encap: VXLAN

203.0.113.12/32, ubest/mbest: 1/0
    *via 198.51.100.10%default, [200/0], 00:13:46, bgp-100, internal, tag 100 (evpn) segid:
 50000 tunnelid: 0x16020202 encap: VXLAN
```

Command output description

Direct means that the subnet prefix is configured locally under a Layer-3 interface on this switch. Local means the IP address belongs to the switch aka locally configured under a Layer-3 interface on that switch (200.0.0.1/24).

The following sample output displays MAC - IP address binding for all attached and remote end hosts (learned through the BGP EVPN control plane):

```
switch # show l2route evpn mac-ip all

Topology    Mac Address    Prod   Flags         Seq No     Host IP         Next-Hops
----------- -------------- ------ ---------- --------------- ---------------
200         2010.0000.0012 HMM    --            0          203.0.113.5     Local
200         2010.0000.0015 BGP    --            0          203.0.113.8     198.51.100.10
200         002a.6a85.a67c BGP    --            0          203.0.113.12    198.51.100.10
```

ARP Suppression

BGP-EVPN distributes MAC and host IP information for hosts below a VTEP. Remote VTEPs can use this information to learn about other hosts and thereby suppress ARP requests early by proxying on behalf of the destination. All the IP-MAC binding aka ARP information learnt either about local end hosts or remote end hosts is shown into an ARP suppression cache. This early ARP termination functionality is enabled on a per Layer 2 VNI basis via a configuration knob (specifically **suppress-arp**). The detailed description of how the suppress-arp function works was described in the **ARP Suppression** section (Chapter 2 - Introducing VXLAN/EVPN). Here we cover the configuration and related show CLIs. This functionality is identical on Cisco Nexus 5000, 7000, 9000 Series switches.

ARP suppression at the VTEP level

(config) #

```
interface nve 1
  source-interface loopback 1
  host-reachability protocol bgp
  member vni 30000
     mcast-group 239.1.1.0
     suppress-arp
```

ARP suppression verification

The following sample output displays ARP suppression information in the cache:

```
switch# show ip arp suppression-cache detail

Flags: + - Adjacencies synced via CFSoE
       L - Local Adjacency
       R - Remote Adjacency
       L2 - Learnt over L2 interface
       PS - Added via L2RIB, Peer Sync
       RO - Dervied from L2RIB Peer Sync Entry

Ip Address      Age     Mac Address    Vlan Physical-ifindex    Flags    Remote Vtep Addrs

203.0.113.5     00:16:01 2010.0000.0012  200 Ethernet1/7         L
203.0.113.12    00:34:28 002a.6a85.a67c  200 (null)              R        198.51.100.10
203.0.113.8     3d00h    2010.0000.0015  200 (null)              R        198.51.100.10
```

The following sample output displays ARP suppression information for a VLAN, in the cache memory:

```
switch# show ip arp suppression-cache vlan 200

Flags: + - Adjacencies synced via CFSoE
       L - Local Adjacency
       R - Remote Adjacency
       L2 - Learnt over L2 interface
       PS - Added via L2RIB, Peer Sync
       RO - Dervied from L2RIB Peer Sync Entry

Ip Address      Age      Mac Address    Vlan Physical-ifindex  Flags   Remote Vtep Addrs

203.0.113.5    00:17:19 2010.0000.0012  200  Ethernet1/7       L
203.0.113.12   00:35:46 002a.6a85.a67c  200  (null)            R        198.51.100.10
203.0.113.8    3d00h    2010.0000.0015  200  (null)            R        198.51.100.10
```

The following sample output displays ARP suppression cache statistics information:

```
switch# show ip arp suppression-cache statistics

ARP packet statistics for suppression-cache
Suppressed:
Total 0, Requests 0, Requests on L2 0, Gratuitous 0, Gratuitous on L2 0

Forwarded :
Total: 0
 L3 mode :      Requests 0, Replies 0
                Request on core port 0, Reply on core port 0
                Dropped 0
 L2 mode :      Requests 0, Replies 0
                Request on core port 0, Reply on core port 0
                Dropped 0

Received:
Total: 3
 L3 mode:       Requests 3, Replies 0
                Local Request 3, Local Responses 0
                Gratuitous 0, Dropped 0
 L2 mode :      Requests 0, Replies 0
```

```
                Gratuitous 0, Dropped 0
ARP suppression-cache Local entry statistics
Adds 3, Deletes 0
```

Unknown unicast (packet) suppression

Configuration example for implementing the *suppress unknown unicast* function on a leaf/ToR switch

(config) #

```
interface nve 1
   member vni 30000
     suppress-unknown-unicast
```

FCoE

FCoE over the VXLAN fabric is not supported. However, FCoE and VXLAN can co-exist. FCoE and VXLAN services are provided on separate ports

To enable FCoE, use separate links from the fabric to MDS and connect to the target device. Refer *Cisco NX-OS FCoE Configuration Guide for Nexus 7000 Series and MDS 9000* and *Cisco Nexus 5600 Series NX-OS Fibre Channel over Ethernet Configuration Guide* for details.

DHCP

Manual and auto configurations of DHCP/DHCPv6 server and client functions on the default VRF, management VRF and non default VRF are given below.

DHCPv4 (clients in the non-default vrf)

With Auto Configuration—Supported scenarios

DHCP server and DHCP client in the same vlan/L2VNI but no DHCP relay on the leaf switch (This results in a Layer-2 flood within the same VLAN/L2 VNI).

DHCP server in the management VRF with the DHCP relay on the leaf switch under an Switch Virtual Interface (SVI) / Bridge Domain Interface (BDI).

Note An SVI is applicable to Cisco Nexus 5600 Series switches and a BDI to Cisco Nexus 7000 Series switches.

DHCP server in the default VRF with DHCP relay on the leaf switch under an SVI/BDI.

Without Auto Configuration—Supported scenarios

DHCP server and DHCP client in the same vlan/L2VNI but no DHCP relay on the leaf switch (so Layer-2 flood within the same VLAN/L2VNI).

DHCP server in the management VRF with DHCP relay on the leaf switch under an SVI/BDI.

DHCP server in the default VRF with the DHCP relay on the leaf switch under an SVI/BDI.

DHCP server and client in a non default VRF with a DHCP relay on the leaf switch under an SVI/BDI.

- For this scenario, you must enable the **advertise-pip** command on the leaf switch (for vPC scenarios).

DHCP server and client in different non default VRFs with a DHCP relay on the leaf switch under an SVI/BDI.

- For this scenario, you must enable the **advertise-pip** command on the leaf switch (for vPC scenarios).

DHCPv6 (clients in the non-default vrf)

Without Auto Configuration—Supported scenarios

DHCPv6 server and DHCPv6 client in the same vlan/L2VNI but no DHCPv6 relay on the leaf switch (so Layer-2 flood within the same VLAN/L2VNI).

DHCP server in the management VRF with DHCP relay on the leaf switch under an SVI/BDI.

DHCP configuration in a vPC setup

When DHCP or DHCPv6 relay function is configured on leaf switches in a vPC setup, and the DHCP server is in the non default, non management VRF, then configure the **advertise-pip** command on the vPC leaf switches. This allows BGP EVPN to advertise Route-type 5 routes with the next-hop using the primary IP address of the VTEP interface

A sample configuration is given below:

(config) #

```
router bgp 100
  address-family l2vpn evpn
    advertise-pip
```

vPC and FEX

The following vPC and FEX (the fabric is extended using Cisco Nexus 2000 Series Fabric Extender device) scenarios are supported:

Cisco Nexus 5600 Series Switches

- FEX, no vPC
- vPC with Active-Active FEX
- vPC with Straight Through FEX
- eVPC or 2-layer vPC with FEX

Refer the FEX link for Cisco Nexus 5600 Series switches. This has the topology and configurations for FEX AA, FEX ST, and FEX ST with vPC scenarios.

Cisco Nexus 7000 Series Switches

- Straight Through FEX connected to the leaf switch.
- vPC with straight through FEX.

In the VXLAN EVPN fabric, the Cisco Nexus 7000 Series switches supports integration with Fabric Extender (FEX) devices.

For detailed conceptual and configuration information for FEX support, refer to *Cisco Nexus 2000 Series Fabric Extender Software Configuration Guide for Cisco Nexus 7000 Series Switches, Release 7.x.*

For detailed vPC information, refer to *Design and Configuration Guide: Best Practices for Virtual Port Channels (vPC) on Cisco Nexus 7000 Series Switches.*

Attention In the VXLAN EVPN fabric, Nexus 7000 Series switches do not support FEX in Active-Active mode.

Note If you configure the vPC peer gateway on the vPC switches, you should not enable the **advertise-pip** command to advertise the primary IP address. The vPC peer gateway and advertise PIP functions are mutually exclusive.

Cisco Nexus 9000 Series Switches

- Straight Through FEX connected to the leaf switch.
- vPC with Straight Through FEX.

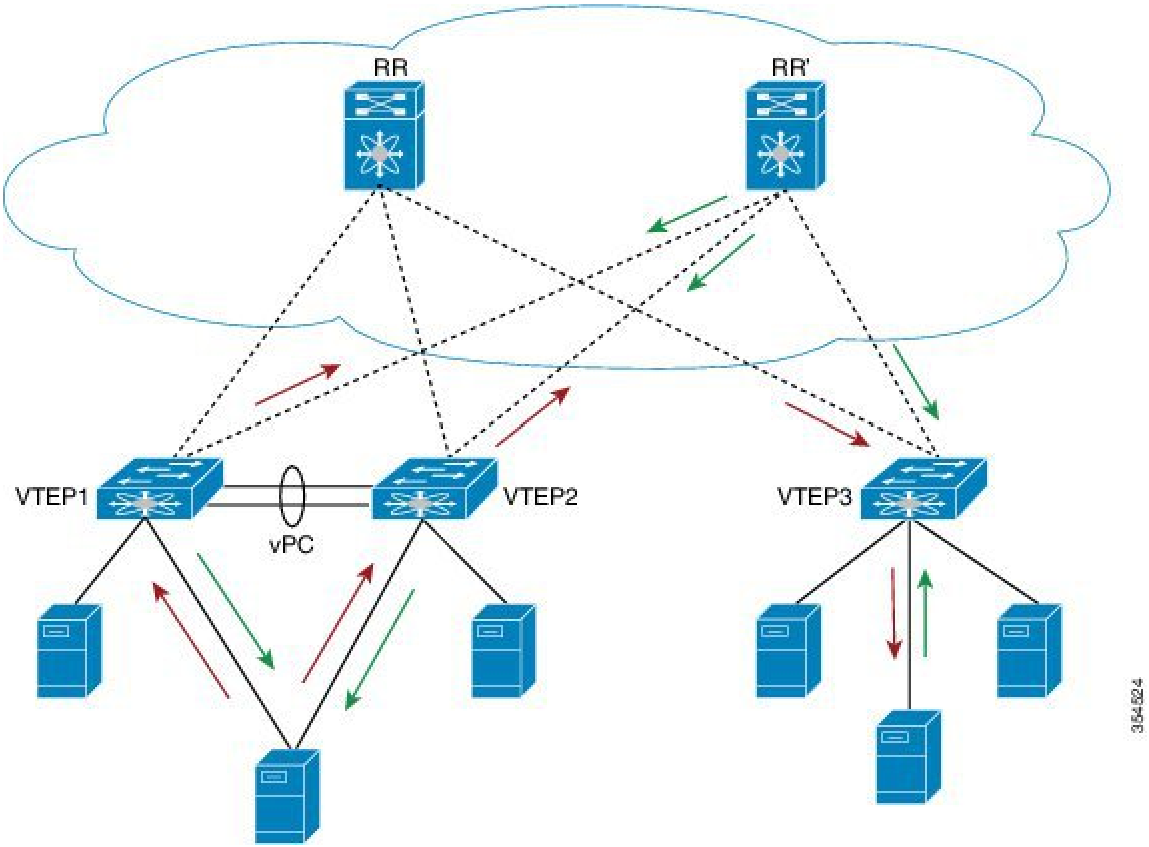
In the VXLAN EVPN fabric, the Cisco Nexus 9000 Series switches supports integration with Fabric Extender (FEX) devices.

For detailed conceptual and configuration information for FEX support, refer to *Cisco Nexus 2000 Series NX-OS Fabric Extender Configuration Guide for Cisco Nexus 9000 Series Switches, Release 7.x*

Attention In the VXLAN EVPN fabric, Nexus 9000 Series switches do not support FEX in Active-Active mode.

vPC configuration for Cisco Nexus 5600 Series Switches

Figure 18: vPC configuration



In the above topology, VTEP 1 and VTEP 2 are ToR switches and vPC peers. Sample vPC configurations are given below. For comprehensive information on vPC, refer to the respective Cisco Nexus Series 5600 and 7000 Series vPC design/configuration guide.

Attention Many of the configurations mentioned below need to be configured identically on the primary and secondary vPC peer switches, noted as **vPC-primary and vPC-secondary peer switches**. Configurations that are different between the peer switches are explicitly mentioned as **vPC-primary peer switch** and **vPC-secondary peer switch**.

Configure the vPC features

(config) #

```
feature lacp
feature vpc
```

Create a vPC domain

vPC-primary peer switch

(config) #

```
vpc domain 100
   peer-keepalive destination 10.1.1.156 source 10.1.1.154
   delay restore 150
   auto-recovery
   ip arp synchronize
   ipv6 nd synchronize
```

vPC-secondary peer switch

(config) #

```
vpc domain 100
  peer-keepalive destination 10.1.1.154 source 10.1.1.156
  delay restore 150
  auto-recovery
  ip arp synchronize
  ipv6 nd synchronize
```

Configure the secondary IP address on the loopback. This will be used as the virtual IP address (vIP) for both vPC peers

The secondary IP address of the source VTEP interface of the fabric (say, VTEP1/VTEP2 as source and VTEP 3 as destination) will be used as the source IP address in the VxLAN outer IP header. In a vPC scenario when EVPN is enabled, EVPN advertises the secondary IP address as the next hop address in the BGP update message. This is true for all route types including MAC routes, host IP routes, prefix routes etc. This is different from VXLAN flood-n-learn operation where for orphan ports the VXLAN outer IP header is set to the physical Peer IP or PIP when traffic ingresses in from the orphan ports and the VIP is only used when traffic ingresses in from the vPC ports.

vPC-primary peer switch

(config) #

```
interface loopback1
  ip address 10.1.2.54/32
  ip address 192.0.2.110/32 secondary

interface nve 1
   source-interface loopback0
   host-reachability protocol bgp
```

vPC-secondary peer switch

(config) #

```
interface loopback1
  ip address 10.1.2.56/32
  ip address 192.0.2.110/32 secondary

interface nve 1
   source-interface loopback1
   host-reachability protocol bgp
```

Note that the secondary IP address configured on the vPC primary and vPC secondary peer switches is the same.

Create the peer-link port-channel

vPC-primary and vPC-secondary peer switches

(config) #

```
interface port-channel 10
  description "vpc-peer-link"
  switchport mode trunk
  spanning-tree port type network
  vpc peer-link
```

Configure the peer-link interface

vPC-primary and vPC-secondary peer switches

(config) #

```
interface Ethernet1/1
  switchport mode trunk
  channel-group 10 mode active
```

Configure the peer link VLAN and routing between the vPC peer switches

Note The **vpc nve peer-link-vlan** command needs to be used only in the Cisco Nexus 5600 Series switches. Cisco Nexus 5600 Series switches encapsulate VXLAN packets over the MCT port with the configured VLAN as the outer-vlan tag while Cisco Nexus 7000,7700,9000 Series switches decapsulate VXLAN packets coming from the core and the decapsulated packet is bridged across the MCT link since they use ASM/SSM protocols.

You can use IS-IS or OSPF as the routing protocol between the vPC peer switches, as mentioned below:

IS-IS

vPC-primary peer switch

(config) #

```
vlan 123
interface Vlan123
  no shutdown
  ip address 38.38.38.54/24
  isis metric 10 level-1
  ip router isis PEER-LINK
  ip pim sparse-mode

vpc nve peer-link-vlan 123
```

vPC-secondary peer switch

(config) #

```
vlan 123
interface Vlan123
  no shutdown
  ip address 38.38.38.56/24
  isis metric 10 level-1
  ip router isis PEER-LINK
  ip pim sparse-mode

vpc nve peer-link-vlan 123
```

OSPF

vPC-primary peer switch

(config) #

```
vlan 123
interface vlan123
 no shutdown
 no ip redirects
 ip address 38.38.38.54/24
 ip ospf cost 10
 ip router ospf PEER-LINK area 0.0.0.0
 ip pim sparse-mode

vpc nve peer-link-vlan 123
```

vPC-secondary peer switch

(config) #

```
vlan 123
interface vlan123
 no shutdown
 no ip redirects
 ip address 38.38.38.56/24
 ip ospf cost 10
 ip router ospf PEER-LINK area 0.0.0.0
 ip pim sparse-mode

vpc nve peer-link-vlan 123
```

Configure the vPC host interface

From the image, you can see that an end host is (dual) attached to the peer switches. You need to configure the peer switches on the same port channel to enable end host dual attachment.

vPC-primary and vPC-secondary peer switches

(config) #

```
interface Ethernet1/5
  switchport mode trunk
  channel-group 35

interface port-channel 35
  switchport mode trunk
  spanning-tree port type edge trunk
```

Exclude the peer link VLAN from server facing ports

vPC-primary and vPC-secondary peer switches

(config) #

```
interface port-channel 35
  switchport trunk allowed vlan except 123

interface e1/5
  switchport trunk allowed vlan except 123
```

BUM (Layer-2 multicast) traffic behavior in VXLAN EVPN environments is identical to that in VXLAN flood and learn environments. For additional information on VXLAN flood and learn, refer to the respective Cisco Nexus Series 5600 or 7000/7700 VXLAN configuration guide.

Scenarios for advertising the primary IP address as the BGP next hop address in a vPC setup

In certain scenarios in a vPC setup (involving ToR leaf or border leaf switches) in the VXLAN EVPN fabric, you need to enable BGP EVPN to advertise Route-type 5 routes with the next-hop using the primary IP address of the VTEP interface. Recall that this is different from the default behavior where the vPC VIP associated with the VTEP interface is used as the next-hop for all advertised routes (Route-types 2/3/5). The scenarios are:

- The leaf switch and its vPC peer have asymmetric external Layer-3 connections that some IP prefix routes are only reachable from one of the leaf switches, and not from both of them.

  For example, when a pair of border leaf switches that run in vPC mode, and are connected to DCI boxes asymmetrically. (A symmetric topology can become asymmetric due to link failure.)

- A DHCP or DHCPv6 relay is configured on the leaf switch and the DHCP server is in the non default, non management VRF.

- Traffic is enabled between the vPC leaf switch and a remote end host.

  For example, to initiate a ping from the leaf switch's loopback address in a non default VRF to a remote end host.

There are 3 types of Layer-3 routes that can be advertised by BGP EVPN. They are:

1. Local host routes - These routes are leant from the attached servers or hosts.
2. Prefix routes - These routes are learnt via other routing protocol at the leaf, border leaf and border spine switches.
3. Leaf switch generated routes - These routes include interface routes and static routes.

On a vPC enabled leaf or border leaf switch, by default all Layer-3 routes are advertised with the secondary IP address (VIP) of the leaf switch VTEP as the BGP next-hop IP address. Prefix routes and leaf switch generated routes are not synced between vPC leaf switches. Using the VIP as the BGP next-hop for these types of routes can cause traffic to be forwarded to the wrong vPC leaf or border leaf switch and black-holed. The provision to use the primary IP address (PIP) as the next-hop when advertising prefix routes or loopback interface routes in BGP on vPC enabled leaf or border leaf switches allows users to select the PIP as BGP next-hop when advertising these types of routes, so that traffic will always be forwarded to the right vPC enabled leaf or border leaf switch.

The configuration command for advertising the PIP is **advertise-pip**.

A sample configuration is given below:

(config) #

```
router bgp 100
  address-family l2vpn evpn
    advertise-pip
    advertise-system-mac
```

The **advertise-pip** command lets BGP use the PIP as next-hop when advertising prefix routes or leaf generated routes if vPC is enabled. The **advertise-system-mac** command lets BGP advertise Route-type-2 routes that includes VIP and router-mac information. This is needed for solving an issue in EVPN decapsulation on remote leaf switches when PIP is used as next-hop in the BGP advertisement.

Ping from a vPC switch when a PIP is not advertised—If you ping from switch A in a vPC setup (comprising of switches A and B) to a connected device or a remote end host, the common, virtual IP address (VIP) is considered the source IP address, and a successful response to the ping will be sent either to A, or to B. If the response is sent to B, then A (the sender) will not receive it.

As a workaround, create a loopback interface with a unique IP address for each vPC switch, and use the loopback IP address as the source for pinging attached devices or end hosts. Also leak the unique address between the vPC pair to ensure that the (ICMP) response is routed back to the sending vPC switch.

Also, you can use the VXLAN OAM functionality as a workaround.

Verify vPC configuration

For verification of MAC routes, refer these commands:

```
vPC-primary peer switch# show mac address-table dynamic

Legend:
        * - primary entry, G - Gateway MAC, (R) - Routed MAC, O - Overlay MAC
        age - seconds since last seen,+ - primary entry using vPC Peer-Link
   VLAN     MAC Address      Type      age     Secure NTFY   Ports/SWID.SSID.LID
---------+-----------------+--------+---------+------+----+------------------
* 200      002a.6a44.9381    dynamic   1800       F    F  Po35
* 200      2010.0000.0010    dynamic   700        F    F  Eth100/1/1
+ 200      2010.0000.0011    dynamic   0          F    F  nve1/10.1.1.56
* 200      2010.0000.0012    dynamic   0          F    F  nve1/10.1.1.74
+ 200      2010.0000.0013    dynamic   0          F    F  nve1/10.1.1.56
* 123      002a.6ab2.0181    dynamic   0          F    F  Po10
* 1        a036.9f19.8ee4    dynamic   0          F    F  Po10
+ 1        a036.9f1a.b970    dynamic   0          F    F  Po10
* 1        a036.9f1a.c134    dynamic   0          F    F  Po10
* 1        a036.9f1a.c135    dynamic   120        F    F  Eth100/1/3
+ 1        a036.9f22.a30e    dynamic   0          F    F  Po10

vPC-secondary peer switch# show mac address-table dynamic

Legend:
        * - primary entry, G - Gateway MAC, (R) - Routed MAC, O - Overlay MAC
        age - seconds since last seen,+ - primary entry using vPC Peer-Link
   VLAN     MAC Address      Type      age     Secure NTFY   Ports/SWID.SSID.LID
---------+-----------------+--------+---------+------+----+------------------
```

```
* 200      002a.6a44.9381    dynamic   300        F    F  Po35
* 200      2010.0000.0010    dynamic   30         F    F  Eth100/1/1
* 200      2010.0000.0011    dynamic   40         F    F  Eth101/1/1
* 200      2010.0000.0012    dynamic   0          F    F  nve1/10.1.1.74
* 200      2010.0000.0013    dynamic   20         F    F  Eth1/6
* 123      002a.6a6e.cbc1    dynamic   0          F    F  Po10
* 1        a036.9f19.8ee4    dynamic   0          F    F  Eth101/1/4
* 1        a036.9f1a.b970    dynamic   1770       F    F  Eth101/1/1
* 1        a036.9f1a.c134    dynamic   30         F    F  Eth101/1/3
* 1        a036.9f1a.c135    dynamic   110        F    F  Eth100/1/3

vPC-primary peer switch# show l2route evpn mac all

Topology    Mac Address    Prod   Next Hop (s)
----------- -------------- ------ ---------------
200         002a.6a44.9381 Local  Po35
200         2010.0000.0010 Local  Eth100/1/1
200         2010.0000.0011 Local  nve1/10.1.1.56
200         2010.0000.0012 BGP    10.1.1.74
200         2010.0000.0013 Local  nve1/10.1.1.56
2200        8c60.4f14.2efc VXLAN  10.1.1.74

vPC-secondary peer switch# show l2route evpn mac all

Topology    Mac Address    Prod   Next Hop (s)
----------- -------------- ------ ---------------
200         002a.6a44.9381 Local  Po35
200         2010.0000.0010 Local  Eth100/1/1
200         2010.0000.0011 Local  Eth101/1/1
200         2010.0000.0012 BGP    10.1.1.74
200         2010.0000.0013 Local  Eth1/6
2200        8c60.4f14.2efc VXLAN  10.1.1.74

vPC-primary peer switch# show bgp l2vpn evpn

BGP routing table information for VRF default, address family L2VPN EVPN
BGP table version is 410, local router ID is 10.1.1.54
Status: s-suppressed, x-deleted, S-stale, d-dampened, h-history, *-valid, >-best
Path type: i-internal, e-external, c-confed, l-local, a-aggregate, r-redist, I-i
njected
Origin codes: i - IGP, e - EGP, ? - incomplete, | - multipath, & - backup

   Network            Next Hop            Metric     LocPrf     Weight Path
Route Distinguisher: 10.1.1.54:32967    (L2VNI 30200)
*>l[2]:[0]:[0]:[48]:[002a.6a44.9381]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100      32768 i
* i                   2.2.2.2                           100          0 i
*>l[2]:[0]:[0]:[48]:[2010.0000.0010]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100      32768 i
* i                   2.2.2.2                           100          0 i
*>l[2]:[0]:[0]:[48]:[2010.0000.0011]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100      32768 i
* i                   2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
*>l[2]:[0]:[0]:[48]:[2010.0000.0013]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100      32768 i
* i                   2.2.2.2                           100          0 i
*>l[2]:[0]:[0]:[48]:[002a.6a44.9381]:[32]:[200.0.0.35]/272
                      2.2.2.2                           100      32768 i
* i                   2.2.2.2                           100          0 i
* i[2]:[0]:[0]:[48]:[2010.0000.0010]:[32]:[200.0.0.10]/272
```

```
                      2.2.2.2                           100          0 i
*>l                   2.2.2.2                           100      32768 i
*>l[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[200.0.0.11]/272
                      2.2.2.2                           100      32768 i
* i                   2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[200.0.0.12]/272
                      10.1.1.74                          100          0 i
*>l[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[200.0.0.13]/272
                      2.2.2.2                           100      32768 i
* i                   2.2.2.2                           100          0 i

Route Distinguisher: 10.1.1.56:3
*>i[5]:[0]:[0]:[24]:[200.0.0.0]:[0.0.0.0]/224
                      2.2.2.2                  0        100          0 ?

Route Distinguisher: 10.1.1.56:32967
*>i[2]:[0]:[0]:[48]:[002a.6a44.9381]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0010]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[002a.6a44.9381]:[32]:[200.0.0.35]/272
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0010]:[32]:[200.0.0.10]/272
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[200.0.0.11]/272
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[200.0.0.13]/272
                      2.2.2.2                           100          0 i

Route Distinguisher: 10.1.1.74:32967
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[200.0.0.12]/272
                      10.1.1.74                          100          0 i
```

```
vPC-secondary peer switch# show bgp l2vpn evpn

BGP routing table information for VRF default, address family L2VPN EVPN
BGP table version is 308, local router ID is 10.1.1.56
Status: s-suppressed, x-deleted, S-stale, d-dampened, h-history, *-valid, >-best
Path type: i-internal, e-external, c-confed, l-local, a-aggregate, r-redist, I-i
njected
Origin codes: i - IGP, e - EGP, ? - incomplete, | - multipath, & - backup

   Network            Next Hop            Metric     LocPrf     Weight Path

Route Distinguisher: 10.1.1.54:3
*>i[5]:[0]:[0]:[24]:[200.0.0.0]:[0.0.0.0]/224
                      2.2.2.2                  0        100          0 ?
Route Distinguisher: 10.1.1.54:32967
*>i[2]:[0]:[0]:[48]:[002a.6a44.9381]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0010]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[002a.6a44.9381]:[32]:[200.0.0.35]/272
                      2.2.2.2                           100          0 i
```

```
*>i[2]:[0]:[0]:[48]:[2010.0000.0010]:[32]:[200.0.0.10]/272
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[200.0.0.11]/272
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[200.0.0.13]/272
                      2.2.2.2                           100          0 i

Route Distinguisher: 10.1.1.56:32967    (L2VNI 30200)
* i[2]:[0]:[0]:[48]:[002a.6a44.9381]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>l                   2.2.2.2                           100      32768 i
* i[2]:[0]:[0]:[48]:[2010.0000.0010]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>l                   2.2.2.2                           100      32768 i
* i[2]:[0]:[0]:[48]:[2010.0000.0011]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>l                   2.2.2.2                           100      32768 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
* i[2]:[0]:[0]:[48]:[2010.0000.0013]:[0]:[0.0.0.0]/216
                      2.2.2.2                           100          0 i
*>l                   2.2.2.2                           100      32768 i
* i[2]:[0]:[0]:[48]:[002a.6a44.9381]:[32]:[200.0.0.35]/272
                      2.2.2.2                           100          0 i
*>l                   2.2.2.2                           100      32768 i
*>l[2]:[0]:[0]:[48]:[2010.0000.0010]:[32]:[200.0.0.10]/272
                      2.2.2.2                           100      32768 i
* i                   2.2.2.2                           100          0 i
* i[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[200.0.0.11]/272

*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[200.0.0.12]/272
                      10.1.1.74                          100          0 i
* i[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[200.0.0.13]/272
                      2.2.2.2                           100          0 i
*>l                   2.2.2.2                           100      32768 i

Route Distinguisher: 10.1.1.74:32967
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[0]:[0.0.0.0]/216
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[200.0.0.12]/272
                      10.1.1.74                          100          0 i
```

```
vPC-primary peer switch# show nve peers

Interface Peer-IP          State LearnType Uptime   Router-Mac
--------- ---------------  ----- --------- -------- -----------------
nve1      10.1.1.56         Up    CP        01:41:24 n/a
nve1      10.1.1.74         Up    CP        01:41:19 8c60.4f14.2efc
```

```
vPC-secondary peer switch# show nve peers

Interface Peer-IP          State LearnType Uptime   Router-Mac
--------- ---------------  ----- --------- -------- -----------------
nve1      10.1.1.54         Up    CP        1d01h    n/a
nve1      10.1.1.74         Up    CP        4d09h    8c60.4f14.2efc
```

For verification of IP host and prefix routes, refer these commands

```
vPC-primary peer switch# show ip arp vrf all

Flags: * - Adjacencies learnt on non-active FHRP router
       + - Adjacencies synced via CFSoE
       # - Adjacencies Throttled for Glean
       D - Static Adjacencies attached to down interface

IP ARP Table for all contexts
Total number of entries: 8
Address         Age       MAC Address     Interface
10.1.1.156      00:13:04  002a.6ab2.0141  mgmt0
10.1.1.233      00:00:23  0050.569f.6c61  mgmt0
1.1.1.53        00:12:51  002a.6a85.a5bc  Ethernet1/24
38.38.38.56     00:02:55  002a.6ab2.0181  Vlan123
200.0.0.10      00:09:02  2010.0000.0010  Vlan200
200.0.0.11      00:06:37  2010.0000.0011  Vlan200          +
200.0.0.13      00:06:34  2010.0000.0013  Vlan200          +
200.0.0.35      00:00:28  002a.6a44.9381  Vlan200          +


vPC-secondary peer switch# show ip arp vrf all

Flags: * - Adjacencies learnt on non-active FHRP router
       + - Adjacencies synced via CFSoE
       # - Adjacencies Throttled for Glean
       D - Static Adjacencies attached to down interface

IP ARP Table for all contexts
Total number of entries: 8
Address         Age       MAC Address     Interface
10.1.1.154      00:13:11  002a.6a6e.cb81  mgmt0
10.1.1.233      00:00:30  0050.569f.6c61  mgmt0
1.1.1.53        00:04:27  002a.6a85.a5bc  Ethernet1/26
38.38.38.54     00:03:03  002a.6a6e.cbc1  Vlan123
200.0.0.10      00:09:09  2010.0000.0010  Vlan200          +
200.0.0.11      00:06:45  2010.0000.0011  Vlan200
200.0.0.13      00:06:41  2010.0000.0013  Vlan200
200.0.0.35      00:00:36  002a.6a44.9381  Vlan200


vPC-primary peer switch# show ip route vrf all

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>

1.1.1.53/32, ubest/mbest: 1/0
    *via 1.1.1.53, Eth1/24, [110/5], 16:40:48, ospf-UNDERLAY, intra
10.1.1.54/32, ubest/mbest: 2/0, attached
    *via 10.1.1.54, Lo0, [0/0], 01:59:11, local
    *via 10.1.1.54, Lo0, [0/0], 01:59:11, direct
10.1.1.56/32, ubest/mbest: 1/0
    *via 1.1.1.53, Eth1/24, [110/9], 16:40:42, ospf-UNDERLAY, intra
10.1.1.74/32, ubest/mbest: 1/0
    *via 1.1.1.53, Eth1/24, [110/9], 16:40:39, ospf-UNDERLAY, intra
2.2.2.2/32, ubest/mbest: 2/0, attached
    *via 2.2.2.2, Lo0, [0/0], 01:59:11, local
    *via 2.2.2.2, Lo0, [0/0], 01:59:11, direct
10.254.254.2/32, ubest/mbest: 1/0
    *via 1.1.1.53, Eth1/24, [110/5], 16:40:00, ospf-UNDERLAY, intra
10.254.254.66/32, ubest/mbest: 1/0
    *via 1.1.1.53, Eth1/24, [110/5], 16:39:55, ospf-UNDERLAY, intra
38.38.38.0/24, ubest/mbest: 1/0, attached
    *via 38.38.38.54, Vlan123, [0/0], 01:59:00, direct
38.38.38.54/32, ubest/mbest: 1/0, attached
    *via 38.38.38.54, Vlan123, [0/0], 01:59:00, local
```

```
IP Route Table for VRF "management"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

0.0.0.0/0, ubest/mbest: 1/0
    *via 10.1.1.233, [1/0], 4d08h, static
10.1.1.0/24, ubest/mbest: 1/0, attached
    *via 10.1.1.154, mgmt0, [0/0], 4d08h, direct
10.1.1.154/32, ubest/mbest: 1/0, attached
    *via 10.1.1.154, mgmt0, [0/0], 4d08h, local

IP Route Table for VRF "sml:vpn2200"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

200.0.0.0/24, ubest/mbest: 1/0, attached
    *via 200.0.0.1, Vlan200, [0/0], 01:59:00, direct, tag 12345,
200.0.0.1/32, ubest/mbest: 1/0, attached
    *via 200.0.0.1, Vlan200, [0/0], 01:59:00, local, tag 12345,
200.0.0.10/32, ubest/mbest: 1/0, attached
    *via 200.0.0.10, Vlan200, [190/0], 01:46:41, hmm
200.0.0.11/32, ubest/mbest: 1/0, attached
    *via 200.0.0.11, Vlan200, [190/0], 01:46:41, hmm
200.0.0.12/32, ubest/mbest: 1/0
    *via 10.1.1.74%default, [200/0], 01:59:04, bgp-100, internal, ta
-vpn)segid 32200 tunnel: 16843082 encap: 1

200.0.0.13/32, ubest/mbest: 1/0, attached
    *via 200.0.0.13, Vlan200, [190/0], 01:46:36, hmm
200.0.0.35/32, ubest/mbest: 1/0, attached
    *via 200.0.0.35, Vlan200, [190/0], 01:58:35, hmm
```

```
vPC-secondary peer switch# show ip route vrf all

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>

1.1.1.53/32, ubest/mbest: 1/0
    *via 1.1.1.53, Eth1/26, [110/5], 16:45:46, ospf-UNDERLAY, intra
10.1.1.54/32, ubest/mbest: 1/0
    *via 1.1.1.53, Eth1/26, [110/9], 02:04:15, ospf-UNDERLAY, intra
10.1.1.56/32, ubest/mbest: 2/0, attached
    *via 10.1.1.56, Lo0, [0/0], 4d10h, local
    *via 10.1.1.56, Lo0, [0/0], 4d10h, direct
10.1.1.74/32, ubest/mbest: 1/0
    *via 1.1.1.53, Eth1/26, [110/9], 16:45:43, ospf-UNDERLAY, intra
2.2.2.2/32, ubest/mbest: 2/0, attached
    *via 2.2.2.2, Lo0, [0/0], 4d10h, local
    *via 2.2.2.2, Lo0, [0/0], 4d10h, direct
10.254.254.2/32, ubest/mbest: 1/0
    *via 1.1.1.53, Eth1/26, [110/5], 16:45:04, ospf-UNDERLAY, intra
10.254.254.66/32, ubest/mbest: 1/0
    *via 1.1.1.53, Eth1/26, [110/5], 16:44:59, ospf-UNDERLAY, intra
38.38.38.0/24, ubest/mbest: 1/0, attached
    *via 38.38.38.56, Vlan123, [0/0], 02:04:13, direct
38.38.38.56/32, ubest/mbest: 1/0, attached
    *via 38.38.38.56, Vlan123, [0/0], 02:04:13, local
```

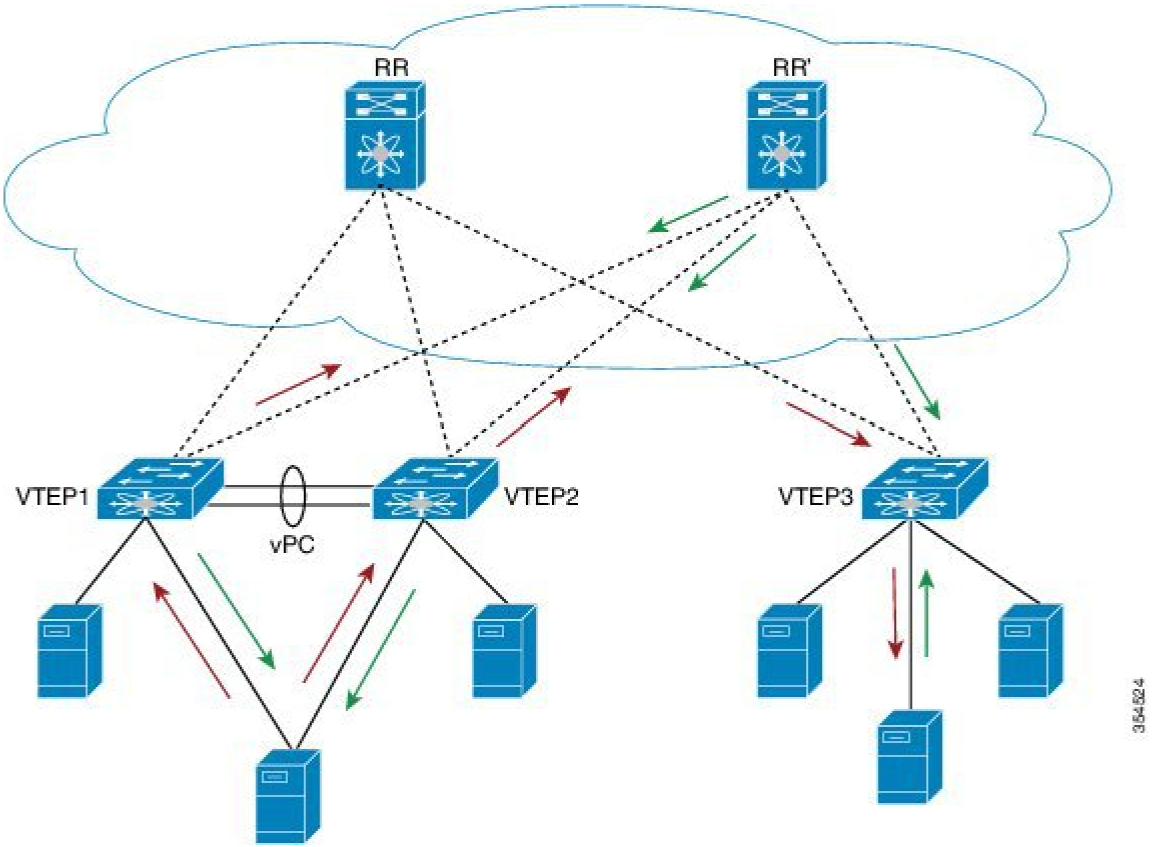
```
IP Route Table for VRF "management"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

0.0.0.0/0, ubest/mbest: 1/0
    *via 10.1.1.233, [1/0], 4d10h, static
10.1.1.0/24, ubest/mbest: 1/0, attached
    *via 10.1.1.156, mgmt0, [0/0], 4d10h, direct
10.1.1.156/32, ubest/mbest: 1/0, attached
    *via 10.1.1.156, mgmt0, [0/0], 4d10h, local

IP Route Table for VRF "sml:vpn2200"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

200.0.0.0/24, ubest/mbest: 1/0, attached
    *via 200.0.0.1, Vlan200, [0/0], 4d10h, direct, tag 12345,
200.0.0.1/32, ubest/mbest: 1/0, attached
    *via 200.0.0.1, Vlan200, [0/0], 4d10h, local, tag 12345,
200.0.0.10/32, ubest/mbest: 1/0, attached
    *via 200.0.0.10, Vlan200, [190/0], 01:51:46, hmm
200.0.0.11/32, ubest/mbest: 1/0, attached
    *via 200.0.0.11, Vlan200, [190/0], 01:51:46, hmm
200.0.0.12/32, ubest/mbest: 1/0
    *via 10.1.1.74%default, [200/0], 02:07:28, bgp-100, internal, tag 100,  (mpls
-vpn)segid 32200 tunnel: 16843082 encap: 1

200.0.0.13/32, ubest/mbest: 1/0, attached
    *via 200.0.0.13, Vlan200, [190/0], 01:51:40, hmm
200.0.0.35/32, ubest/mbest: 1/0, attached
    *via 200.0.0.35, Vlan200, [190/0], 02:03:39, hmm
```

```
vPC-primary peer switch# show l2route evpn mac-ip all

Topology ID Mac Address    Prod Host IP                                 Next Hop
 (s)
----------- -------------- ---- --------------------------------------- --------
200         002a.6a44.9381 HMM  200.0.0.35                              N/A

200         2010.0000.0012 BGP  200.0.0.12                              10.1.1.74


vPC-secondary peer switch# show l2route evpn mac-ip all

n6k-56-poap# show l2route evpn mac-ip all
Topology ID Mac Address    Prod Host IP                                 Next Hop
 (s)
----------- -------------- ---- --------------------------------------- --------
200         002a.6a44.9381 HMM  200.0.0.35                              N/A

200         2010.0000.0012 BGP  200.0.0.12                              10.1.1.74


vPC-primary peer switch# show bgp l2vpn evpn

Route Distinguisher: 10.1.1.54:3    (L3VNI 32200)
*>i[2]:[0]:[0]:[48]:[002a.6a44.9381]:[32]:[200.0.0.35]/272
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0010]:[32]:[200.0.0.10]/272
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[200.0.0.11]/272
```

```
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[200.0.0.12]/272
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[200.0.0.13]/272
                      2.2.2.2                           100          0 i
*>l[5]:[0]:[0]:[24]:[200.0.0.0]:[0.0.0.0]/224
                      2.2.2.2                  0        100      32768 ?
* i                   2.2.2.2                  0        100          0 ?

vPC-secondary peer switch# show bgp l2vpn evpn

Route Distinguisher: 10.1.1.56:3    (L3VNI 32200)
*>i[2]:[0]:[0]:[48]:[002a.6a44.9381]:[32]:[200.0.0.35]/272
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0010]:[32]:[200.0.0.10]/272
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0011]:[32]:[200.0.0.11]/272
                      2.2.2.2                           100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0012]:[32]:[200.0.0.12]/272
                      10.1.1.74                          100          0 i
*>i[2]:[0]:[0]:[48]:[2010.0000.0013]:[32]:[200.0.0.13]/272
                      2.2.2.2                           100          0 i
* i[5]:[0]:[0]:[24]:[200.0.0.0]:[0.0.0.0]/224
                      2.2.2.2                  0        100          0 ?
*>l                   2.2.2.2                  0        100      32768 ?
```

vPC configuration for Cisco Nexus 9000 Series Switches

Figure 19: vPC configuration



Configure the vPC features

(config) #

```
feature vpc
```

Create a vPC domain

vPC-primary peer switch

(config) #

```
vpc domain 1
  peer-switch
  peer-keepalive destination 192.0.2.30 source 192.0.2.20
  delay restore 150
  peer-gateway
  auto-recovery
  ip arp synchronize
  ipv6 nd synchronize
```

vPC-secondary peer switch

(config) #

```
vpc domain 1
 peer-switch
 peer-keepalive destination 192.0.2.30 source 192.0.2.20
 delay restore 150 peer-gateway
 auto-recovery
 ip arp synchronize
 ipv6 nd synchronize
```

Configure the secondary IP address on the loopback. This will be used as the virtual IP address (vIP) for both vPC peers

The secondary IP address of the source VTEP interface on vPC leaf switches will be used as the source IP address in the VXLAN outer IP header. In a vPC scenario when EVPN is enabled, EVPN advertises the secondary IP address as the next hop address in the BGP update message. This is true for all route types including MAC routes, host IP routes, prefix routes, etc.

vPC-primary peer switch

(config) #

```
interface loopback1
  ip address 192.0.2.40/32
  ip address 198.51.100.10/32 secondary
  ip router isis UNDERLAY
  ip pim sparse-mode

interface nve 1
  source-interface loopback1
  host-reachability protocol bgp
```

vPC-secondary peer switch

(config) #

```
interface loopback1
  ip address 192.0.2.41/32
  ip address 198.51.100.10/32 secondary
  ip router isis UNDERLAY
  ip pim sparse-mode

interface nve 1
  source-interface loopback1
  host-reachability protocol bgp
```

Note that the secondary IP address configured on the vPC primary and vPC secondary peer switches is the same.

Create the peer-link port-channel

vPC-primary and vPC-secondary peer switches

(config) #

```
interface port-channel 10
  description "vpc-peer-link”
  switchport mode trunk
  spanning-tree port type network
  vpc peer-link
```

Configure the peer-link interface

vPC-primary and vPC-secondary peer switches

(config) #

```
interface Ethernet1/1
  description “vpc-peer-link”
  switchport mode trunk
  channel-group 10 mode active
```

Configure the backup VLAN path between vPC peer switches

Note To provide a backup path when a vPC switch loses connectivity to the spine, at least one SVI is required to be configured across the peer-link, so that traffic can be forwarded to this vPC switch from its vPC peer switch over the peer-link.

You can use IS-IS or OSPF as the routing protocol between the vPC peer switches, as mentioned below:

IS-IS

vPC-primary peer switch

(config) #

```
vlan 123
interface Vlan123
  no shutdown
  ip address 192.0.2.100/24
  ip router isis UNDERLAY
  ip pim sparse-mode
  no ip redirects
  no ipv6 redirects

system nve infra-vlan 123
```

vPC-secondary peer switch

(config) #

```
vlan 123
interface Vlan123
  no shutdown
  ip address 192.0.2.101/24
  ip router isis UNDERLAY
  ip pim sparse-mode
  no ip redirects
  no ipv6 redirects

system nve infra-vlan 123
```

OSPF

vPC-primary peer switch

(config) #

```
vlan 123
interface Vlan123
  no shutdown
  ip address 192.0.2.100/24
  ip router ospf UNDERLAY area 0.0.0.0
  ip ospf network point-to-point
  ip pim sparse-mode
  no ip redirects
  no ipv6 redirects

system nve infra-vlan 123
```

vPC-secondary peer switch

(config) #

```
vlan 123 interface vlan123
  no shutdown
  ip address 192.0.2.100/24
  ip router ospf UNDERLAY area 0.0.0.0
  ip ospf network point-to-point
  ip pim sparse-mode
  no ip redirects
  no ipv6 redirects

system nve infra-vlan 123
```

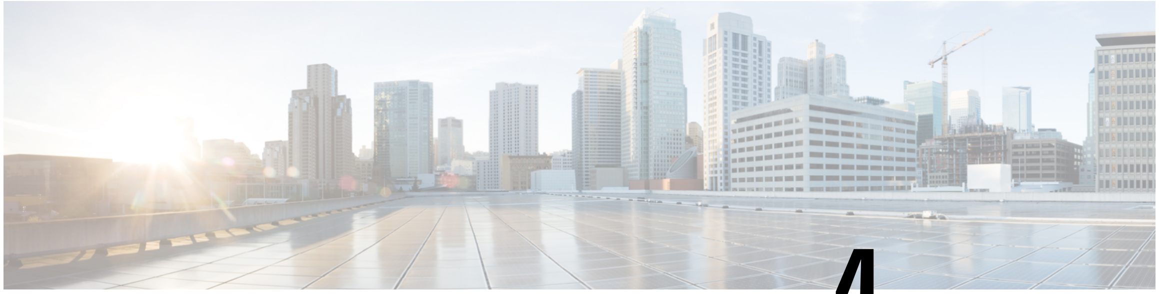
Configure the vPC host interface

As shown in the figure, an end host is (dual) attached to both vPC peer switches. Same port channel must be configured on both switches to enable end host dual attachment.

vPC-primary and vPC-secondary peer switches

(config) #

```
interface Ethernet1/2
  switchport mode trunk
  channel-group 52

interface port-channel 52
  switchport mode trunk
  vpc 52
```



CHAPTER 4

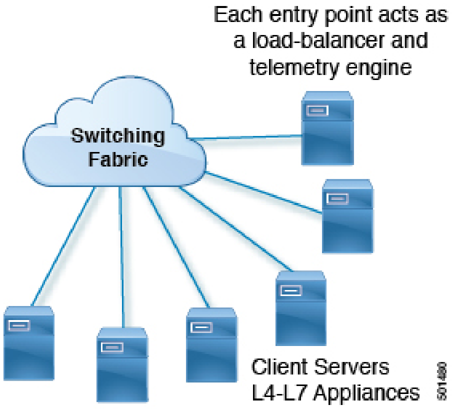
Pervasive Load Balancing

Pervasive Load Balancing for the Programmable Fabric

In a programmable fabric, the servers, the virtual machines (VMs), and the containers (specific to a given service) can be distributed across the fabric, and attached to different ToR or leaf switches. The Pervasive Load Balancing (PLB) feature enables load balancing to the servers that are distributed across the fabric.

PLB enables fabric to act as a massive load-balancer and makes it capable of providing massive telemetry and analytics. When PLB is used as a load-balancer, you can connect between Layer 4 and Layer 7 appliances anywhere in the fabric. This is shown in figure, *Load Balancing across the Fabric*.

Figure 20: Load Balancing across the Fabric


Each entry point acts as a load-balancer and telemetry engine
Switching Fabric
Client Servers
L4-L7 Appliances
501480

You may have a large number of clients (local and across the border leaf), that include database servers, application servers, web servers, firewalls, WAAS, IPS, IDS, and video caches. The information about traffic flowing to each firewall, WAAS, IPS, IDS, and server from each device in the fabric, including information about when traffic is high or low is very valuable.

PLB sits on the path between clients and servers or Layer 4 and Layer 7 services, making PLB aware about traffic information. With this information PLB provides valuable traffic analytics and telemetry.

In the load balancing function, a virtual IP (VIP) abstracts a service provided by a physical server farm distributed across the DC fabric. When different clients (local to fabric or from a remote location) send requests for a given service, these requests are always destined to the VIP of these servers.

On the ToR or leaf switches, PLB matches the source IP address bits and mask, the destination IP address (Virtual IP address), and relevant Layer 3 or Layer 4 fields to load balance these requests among the servers.

PLB provides an infrastructure to configure a cluster of the servers (nodes) inside a device group. It segregates the client traffic based on the buckets (bit mask), and the tenant SVI configured under the PLB service. Based on the defined cluster of nodes (servers) and buckets, PLB automatically creates rules to match the client IP traffic into the buckets mask and redirects the matched traffic to a specific server node.

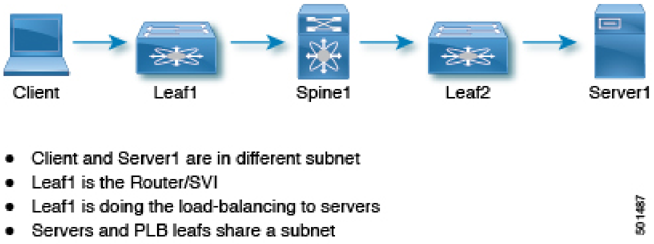
PLB provides the infrastructure to configure cluster of servers' nodes inside a device group and segregates client traffic based on the buckets (bit mask) and tenant SVI configured under PLB service. Based on the defined cluster of nodes (servers) and buckets.

PLB also provides the infrastructure to periodically monitor health of all server nodes and status of its application services such as TCP, UDP, and DNS on a given VRF.

In case, if server become non-responsive or non-operational then PLB automatically switches the client traffic from the non-operational node to a single or group of configured standby nodes. Traffic assignment is achieved by automatically changing flows to a standby node.

PLB currently uses Direct Server Return (DSR) concept and functionality so that server responses are directly sent to the client.

Figure 21: Direct Server Return


Client
Leaf1
Spine1
Leaf2
Server1
Client and Server1 are in different subnet
Leaf1 is the Router/SVI
Leaf1 is doing the load-balancing to servers
Servers and PLB leafs share a subnet
501487

PLB is fabric agnostic but currently supported with VXLAN EVPN Fabric.

PLB is currently supported on Cisco Nexus 9000 Series switches that support PBR over VXLAN.

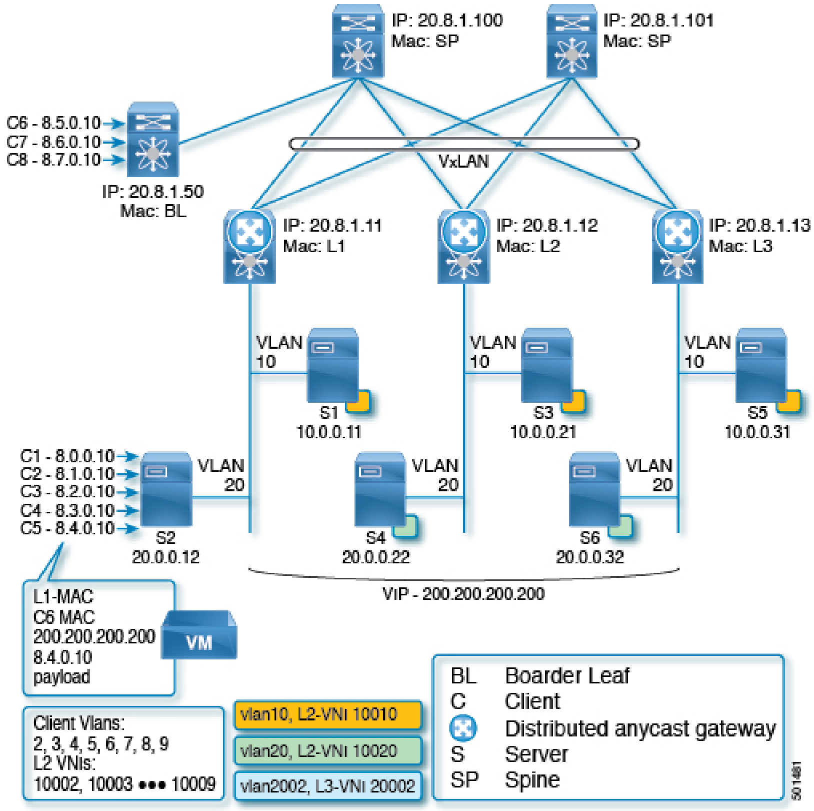
High-level Overview of Designing PLB Topology

A high-level overview of designing Pervasive Load Balancing on the ToR switch is as follows:

- Identify load balancing servers and create a device group.
- Create a PLB service instance for the group, and complete the following:
    - Associate a virtual IP address (VIP) for incoming PLB traffic. The VIP represents the servers in the device group.
    - Enable other load balancing configurations.

The figure, *An Example of PLB Topology*, illustrates how PLB load balances east-west and north-south data center client traffic to multiple servers distributed across the fabric for a VIP.

Figure 22: An Example of PLB Topology



PLB is achieved at line-rate speed, with which, it also provides health monitoring and fail action handling capabilities.

Configuring PLB on Cisco Nexus 9000 Series Switches

Configuring PLB

Use the **feature plb** CLI to enable Pervasive Load Balancing (PLB) under the global configuration mode. Before configuring the PLB device group (cluster of servers) and services, PLB should be enabled in the system.

Note The **feature pbr** and **feature sla sender** CLIs are the prerequisites for configuring PLB.

Sample Configuration

```
switch(config)# feature plb
switch# show feature | grep plb
plb                    1          enabled
```

The **no feature plb** CLI removes the PLB configuration from the system. By default, PLB is disabled in the system.

Configuring PLB Device Group

You can configure the PLB device groups using the **plb device-group** <*dg-name*> CLI. All the nodes (servers) are configured under a device group submode.

Sample Configuration

```
switch(config)# plb device-group dg200
switch(config-device-group)#
switch# show running-config plb-services

!Command: show running-config plb-services
!Time: Mon Jul 24 11:48:10 2017

version 7.0(3)I6(1)
feature plb

plb device-group dg200
```

The **no plb device-group** <*dg-name*> CLI removes the PLB device group from the configuration.

Note A specified device-group configuration cannot be removed if it is being used under the PLB service. It should be removed from the PLB service first before removing it globally.

Configuring Nodes (Servers)

The nodes (servers clusters) can be configured using the **node ip** <*ip-addr*> CLI under a device-group sub-mode. Maximum 64 nodes can be configured inside a device group.

Sample Configuration

```
switch(config)# plb device-group dg200
switch(config-device-group)# node ip 10.0.0.31
leaf3(config-dg-node)#
switch# show running-config plb-services

!Command: show running-config plb-services
!Time: Mon Mar 27 13:17:55 2017

version 7.0(3)I6(1)
feature plb

plb device-group dg200
   node ip 10.0.0.31
```

Use the **no node ip** <*ip-addr*> CLI to remove the node or server configuration under the device group.

Configuring Standby Nodes

Use the **standby ip** <*ip-addr*> CLI to configure the standby node for an active node.

A node-level standby can be associated for each node.

The standby value specifies the standby node information for this active node.

The standby node for an active node, attached to the local leaf is strongly recommended.

Sample Configuration

```
switch(config)# plb device-group dg200
switch(config-device-group)# node ip 10.0.0.21
switch(config-dg-node)# standby ip 20.0.0.22

switch# show running-config plb-services

!Command: show running-config plb-services
!Time: Mon Mar 27 13:17:55 2017

version 7.0(3)I6(1)

feature plb

plb device-group dg200
   node ip 10.0.0.21
       standby ip 20.0.0.22
   node ip 10.0.0.31
       standby ip 20.0.0.32
```

Use the **no standby ip** <*ip-addr*> CLI to remove the standby node configuration for an active node.

Using the Weight option

Use the **weight** <*value*> CLI to configure the weight for a node.

The weight value keyword specifies the proportionate weight for the node for the weighted traffic distribution.

The weight can be assigned only to the active node.

The default weight is 1 for each node and the maximum value to which it can be configured is 8.

Sample Configuration

```
switch(config)# plb device-group dg200
switch(config-device-group)# node ip 10.0.0.21
switch(config-dg-node)# weight 2
switch# show running-config plb-services

!Command: show running-config plb-services
!Time: Mon Mar 27 13:17:55 2017

version 7.0(3)I6(1)

feature plb

plb device-group dg200
   node ip 10.0.0.21
       standby ip 20.0.0.22
   weight 2
   node ip 10.0.0.31
       standby ip 20.0.0.32
```

Use the **no weight** <*value*> CLI to remove the weight for a node or a server.

Configuring Probe for a Node

Use the **probe** {**icmp**| **tcp port** <*port-number*>| **udp port** <*port-number*>| **dns**{<*hostname*> | <*target-address*>}} [**frequency** <*seconds*>] [[**retry-down-count** | **retry-up-count**] <*number*>] [**timeout** <*seconds*>] CLI to configure probe for a node or a server.

A node-level probe (ICMP/TCP/UDP) can be configured to monitor the health of the node. The probe value specifies the probe parameters to use for monitoring the health of this active node.

Note The **feature sla sender** and **feature sla responder** CLI should be enabled for the probe functionality to be enabled.

The frequency field specifies the interval for the probe. The default frequency is 10 Seconds.

The retry-down-count field specifies the consecutive number of times that the probe must have failed prior to the node being marked as operationally *down*. The default retry-down-count is 3.

The retry-up-count field specifies the consecutive number of times the probe must have succeeded prior to the node being marked as operationally *up*. The default retry-up-count is 3.

The timeout field specifies the number of seconds to wait for the probe response. The default timeout is 5 Seconds.

Sample Configuration

```
switch(config)# plb device-group dg200
switch(config-device-group)# node ip 10.0.0.21
switch(config-dg-node)# probe icmp retry-down-count 3 retry-up-count 3 timeout 60 frequency
 60


switch# show running-config plb-services
!Command: show running-config plb-services
!Time: Mon Mar 27 13:17:55 2017
version 7.0(3)I6(1)

feature plb

plb device-group dg200
  node ip 10.0.0.21
    probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    standby ip 20.0.0.22
      probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    weight 2
  node ip 10.0.0.31
   probe icmp
    standby ip 20.0.0.22
      probe icmp
```

Use the **no probe** {**icmp**| **tcp port** <*port-number*>| **udp port** <*port-number*>| **dns**{<*hostname*> | <*target-address*>}} [**frequency** <*seconds*>] [[**retry-down-count** | **retry-up-count**] <*number*>] [**timeout** <*seconds*>] CLI to remove the probe configuration for a node.

Note When you configure a probe for an active node, you must also configure it for the standby node (if it is present).

Configuring the PLB Service at the Global Level

Configure the PLB service using the **plb** <*service-name*> CLI.

Sample Configuration

```
switch(config)# plb srv200
switch(config-plb)#

switch(config-plb)# show running-config plb-services
!Command: show running-config plb-services
!Time: Mon Mar 27 14:17:43 2017
version 7.0(3)I6(1)

feature plb

plb device-group dg200
  node ip 10.0.0.21
    probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    standby ip 20.0.0.22
      probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    weight 2
  node ip 10.0.0.31
    probe icmp
    standby ip 20.0.0.22
      probe icmp

plb srv200
```

Use the **no plb** <*service-name*> command to remove the PLB service.

Configuring Default Device Group under the PLB Service

Use the **device-group** <*dg-name*> CLI to associate a configured device group to the PLB service.

Sample Configuration

```
switch(config)# plb srv200
switch(config-plb)# device-group dg200

switch(config-plb)# show running-config plb-services
!Command: show running-config plb-services
!Time: Mon Mar 27 14:17:43 2017
version 7.0(3)I6(1)

feature plb

plb device-group dg200
  node ip 10.0.0.21
    probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    standby ip 20.0.0.22
      probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    weight 2
  node ip 10.0.0.31
    probe icmp
    standby ip 20.0.0.22
      probe icmp
plb srv200
  device-group dg200
```

Use the **no device-group** <*dg-name*> CLI to detach a device group from the service.

Configuring Virtual IP (VIP) for the PLB Service

Use the **virtual ip** *<ipv4-address >* *<ipv4-network-mask >*[**tcp** | **udp** {**port-number** | **any**}] [**device-group** *<dg-name>*] CLI to configure virtual IP address for the PLB service for a cluster of nodes.

Same VIP cannot be used with different device groups within the same PLB service.

Maximum 64 VIPs can be configured inside the PLB service.

Note The device group can be associated with a VIP. Using this option, multiple device groups can be a part of one service. The default device- group configuration is not required if the device group is specified with all the VIPs configured under the PLB service.

Sample Configuration

Example 1: Default device group configuration and VIP in separate commands

```
switch(config)# plb srv200
switch(config-plb)# device-group dg200
switch(config-plb)# virtual ip 200.200.200.200 255.255.255.255

switch(config-plb)# show running-config plb-services
!Command: show running-config plb-services
!Time: Mon Mar 27 14:17:43 2017
version 7.0(3)I6(1)

feature plb

plb device-group dg200
  node ip 10.0.0.21
    probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    standby ip 20.0.0.22
      probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    weight 2
  node ip 10.0.0.31
    probe icmp
    standby ip 20.0.0.22
      probe icmp

plb srv200
device-group dg200    → default device group
virtual ip 200.200.200.200 255.255.255.255
```

Example 2: VIP with default device group configuration in a single command

```
switch(config)# plb srv200
switch(config-plb)# virtual ip 200.200.200.201 255.255.255.255 device-group dg201

switch(config-plb)# show running-config plb-services
!Command: show running-config plb-services
!Time: Mon Mar 27 14:17:43 2017
version 7.0(3)I6(1)

feature plb

plb device-group dg201
  node ip 10.0.0.21
```

```
    probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    standby ip 20.0.0.22
probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    weight 2
  node ip 10.0.0.31
    probe icmp
    standby ip 20.0.0.22
      probe icmp

plb srv200
virtual ip 200.200.200.200 255.255.255.255 device-group dg201
```

Note Configure VIP with the approach mentioned in examples 1 or 2 in the sample configuration but do not configure both the sample configurations together.

Configuring VRF under the PLB Service

Use the **vrf** <*vrf-name*> CLI to configure VRF on the PLB service. The VRF configuration is required for the probes to work.

Sample Configuration

```
switch(config)# plb srv200
switch(config-plb)# vrf vpn1

switch(config-plb)# show running-config plb-services
!Command: show running-config plb-services
!Time: Mon Mar 27 14:17:43 2017
version 7.0(3)I6(1)

feature plb

plb device-group dg200
  node ip 10.0.0.21
    probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    standby ip 20.0.0.22
      probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    weight 2
  node ip 10.0.0.31
    probe icmp
    standby ip 20.0.0.22
      probe icmp

plb srv200

vrf vpn1
virtual ip 200.200.200.200 255.255.255.255 device-group dg200
```

Use the **no ingress interface** <*interface*> CLI to remove an interface or interfaces from a service.

Configuring Source Interface under the PLB Service

Use the **source-interface loopback** <*loopback-id*> CLI to configure the source interface for the PLB service.

The source interface configuration is required for the probes on the vPC hosts or when the local VM (servers) moves from the local leaf to the remote leaf.

The prerequisite for configuring the source interface under the PLB service is that the loopback interface that gets assigned to the PLB service should be created first on the same VRF as the PLB VRF (tenant VRF) and the IP address with /32 mask should be assigned prior to using this loopback as the source interface on the PLB service.

Additionally, per VRF IGP session should be created in case of vPC switches, for the vPC host probe to work between the vPC peers.

Note Alternatively, if per VRF IGP session is not created, use **advertise-pip** command under the bgp instance. For more details, see the bgp configuration details.

The loopback and per VRF IGP session configuration examples are mentioned in the sample configuration.

An ingress interface should be in the same VRF as the PLB service.

Sample Configuration

The loopback configuration prior to the source interface configuration:

1. Create a loopback interface as displayed in the example:

```
switch# show running-config interface loopback30
interface loopback30
  vrf member vpn1
  ip address 20.8.1.103/32 tag 12345
  ip pim sparse-mode
```

2. Create regular VLAN and per VRF SVI as displayed in the example: (It is needed only for vPC).

```
switch# show run vlan 3
vlan 3

switch# show running-config interface Vlan3
interface Vlan3
  no shutdown
  vrf member cisco:foo-1
  ip address 11.11.11.11/30 tag 12345 (assign 11.11.11.12 for other VPC peer)
```

3. Create bgp neighbor association between the vPC peer under the bgp instance (It is needed only for vPC).

```
switch#show run bgp
router bgp 65101
..
vrf cisco:foo-1
   ..
    neighbor 11.1.1.2 -> For per VRF IGP session, Ip from Interface Vlan 3.
      remote-as 65101
      address-family ipv4 unicast
```

4. Make sure that the per VRF IGP session becomes operationally *up* (It is needed only for vPC).

```
switch#sh ip bgp vrf cisco:foo-1 summary
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
```

```
   11.1.1.1        4 65101      38      37     1504    0    0 00:31:37 9
```

See the following details for the source-interface configuration under the PLB service:

```
switch(config)# plb srv200
switch(config-plb)# source-interface loopback30
switch(config-plb)# show running-config plb-services
!Command: show running-config plb-services
!Time: Mon Mar 27 14:17:43 2017
version 7.0(3)I6(1)

feature plb

plb device-group dg200
  node ip 10.0.0.21
    probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    standby ip 20.0.0.22
      probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    weight 2
  node ip 10.0.0.31
    probe icmp
    standby ip 20.0.0.22
      probe icmp

plb srv200

vrf vpn1
source-interface loopback30
virtual ip 200.200.200.200 255.255.255.255 device-group dg200
```

Use the **no source-interface loopback** <*id*> CLI to remove an interface or interfaces from a service.

Configuring Ingress Interface Configuration under the PLB Service

Use the **ingress interface** <*interface*> CLI to add an ingress interface or multiple interfaces to the PLB service.

Note An ingress interface should be in the same VRF as the PLB service.

Sample Configuration

```
switch(config)# plb srv200
switch(config-plb)# ingress interface Vlan2
switch(config-plb)# show running-config plb-services

!Command: show running-config plb-services
!Time: Mon Mar 27 14:17:43 2017

version 7.0(3)I6(1)
feature plb

plb device-group dg200
  node ip 10.0.0.21
    probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    standby ip 20.0.0.22
      probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    weight 2
  node ip 10.0.0.31
    probe icmp
    standby ip 20.0.0.22
      probe icmp
plb srv200 vrf vpn1
source-interface loopback30
virtual ip 200.200.200.200 255.255.255.255 device-group dg200
ingress interface Vlan2 -> this is in vrf vpn1
```

Use the **no ingress interface** <*interface*> CLI to remove an interface or interfaces from a service.

Configuring the Load-Balancing Options for the PLB Service

Use the **load-balance** {**method** {**src** {**ip** | **ip-l4port** [**tcp** | **udp**] **range** <*x*> <*y*>} | **dst** {**ip** | **ip-l4port** [**tcp** | **udp**] **range** <*x*> <*y*>}} | **buckets** <*count*> | **mask-position** <*position*>} to configure the load-balancing for the PLB service using the following options:

- Buckets—Specifies the number of buckets to be created. The buckets must be configured in the power of two (If the number is not specified, the system then automatically allocates in the power of 2 based on the number of nodes).
- Mask-position—Specifies the mask position of the load balance, starting of the bucket position (If it is not specified, then the default is 0th position but if it is specified, then it starts from 8th bit of the IP address. If it is configured greater than 23 till 32, then it restarts again from 0th position).
- Method—Specifies the source IP address or the destination IP address, or the source IP address and the source port, or the destination IP address and the destination port based load-balancing (With respect to PLB, the source IP address based load balancing is required).

Sample Configuration

```
switch(config)# plb srv200
switch(config-plb)#   load-balance buckets 2 mask-position 11
switch(config-plb)# show running-config plb-services

!Command: show running-config plb-services
!Time: Mon Mar 27 14:17:43 2017

version 7.0(3)I6(1)
feature plb

plb device-group dg200
  node ip 10.0.0.21
    probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    standby ip 20.0.0.22
      probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    weight 2
  node ip 10.0.0.31
    probe icmp
    standby ip 20.0.0.22
      probe icmp

plb srv200
  device-group dg200
  virtual ip 200.200.200.200 255.255.255.255
  ingress interface Vlan2
  load-balance buckets 2 mask-position 11
```

Configuring Failaction Node Reassignment for the PLB Service

The **failaction node reassign** CLI configures failaction node reassignment for the PLB service. Failaction for PLB enables the traffic on the failed nodes that can be reassigned to the first available active probe configured node.

Note Make sure that you that probes have been configured on the local or remote (at least 1) node to achieve failaction reassignment. Otherwise, if there are no active probes configured for nodes in the device group, traffic might be affected during failaction in case the reassigned node is not active.

Sample Configuration

```
switch(config)# plb srv200
switch(config-plb)#   failaction node reassign
switch(config-plb)# show running-config plb-services

!Command: show running-config plb-services
!Time: Mon Mar 27 14:17:43 2017

version 7.0(3)I6(1)
feature plb
plb device-group dg200
  node ip 10.0.0.21
    probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    standby ip 20.0.0.22
      probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    weight 2
  node ip 10.0.0.31
    probe icmp
    standby ip 20.0.0.22
      probe icmp

plb srv200
  device-group dg200
  virtual ip 200.200.200.200 255.255.255.255
  ingress interface Vlan2
  load-balance buckets 2 mask-position 11
  failaction node reassign
```

Shutting Down the PLB Service

Use the **shut** command to shutdown the PLB service. Before modifying any of the configuration under the PLB service, service should be shutdown.

By default, the PLB service is in shut mode. Shutting down the PLB service may cause traffic disruption, if the traffic is already flowing.

The **no shut** CLI is used to unshut the PLB service and to enable the PLB functionality.

Sample Configuration

```
switch(config)# plb srv200 switch(config-plb)# no shut
switch(config-plb)# show running-config plb-services
!Command: show running-config plb-services
!Time: Mon Mar 27 14:17:43 2017
version 7.0(3)I6(1)

feature plb
```

```
plb device-group dg200
  node ip 10.0.0.21
    probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    standby ip 20.0.0.22
      probe icmp frequency 60 timeout 60 retry-down-count 3 retry-up-count 3
    weight 2
  node ip 10.0.0.31
    probe icmp
    standby ip 20.0.0.22
      probe icmp

plb srv200
vrf vpn1
source-interface loopback30
virtual ip 200.200.200.200 255.255.255.255 device-group dg200
ingress interface Vlan2
load-balance buckets 2 mask-position 11 failaction node reassign
no shut
```

Enabling PLB Analytics

Use the **plb analytics** <*plb-service-name*> CLI to enable PLB analytics for a given PLB service.

Note If the PLB service is shut down after configuring the **plb analytics** <*plb-service-name*> command, the same command should be configured again to get the PLB analytics.

Sample Configuration

```
switch(config)# plb analytics srv200
```

The figures 6 through 10 show how PLB facilitates analytics information collections with multiple options and combinations.

Figure 23: PLB Analytics for Servers of Device Group DG1

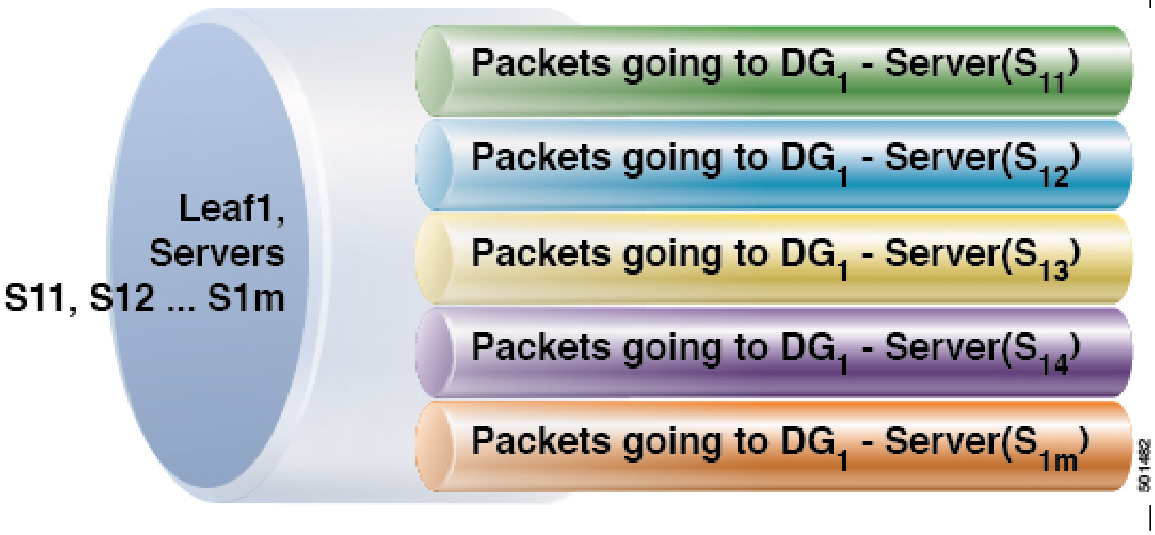

Leaf1, Servers S11, S12 ... S1m
Packets going to DG$_1$ - Server(S$_{11}$)
Packets going to DG$_1$ - Server(S$_{12}$)
Packets going to DG$_1$ - Server(S$_{13}$)
Packets going to DG$_1$ - Server(S$_{14}$)
Packets going to DG$_1$ - Server(S$_{1m}$)
501482

Figure 24: PLB Leaf Analytics for Device Group DG1

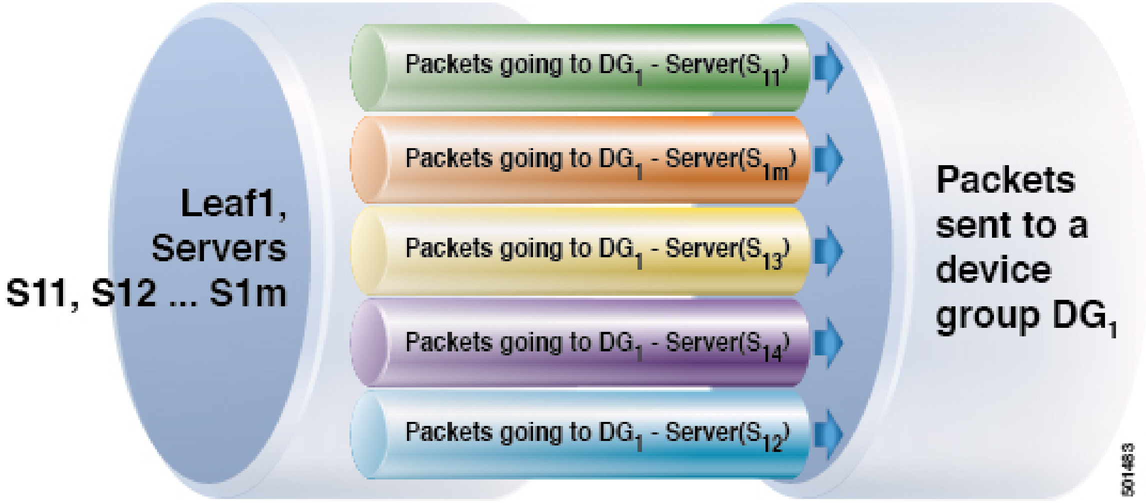

Leaf1,
Servers
S11, S12 ... S1m
Packets going to DG_1 - Server(S_{11})
Packets going to DG_1 - Server(S_{1m})
Packets going to DG_1 - Server(S_{13})
Packets going to DG_1 - Server(S_{14})
Packets going to DG_1 - Server(S_{12})
Packets
sent to a
device
group DG_1
501483

Figure 25: Analytics for Servers of Different Device Groups from Different Tenant SVIs

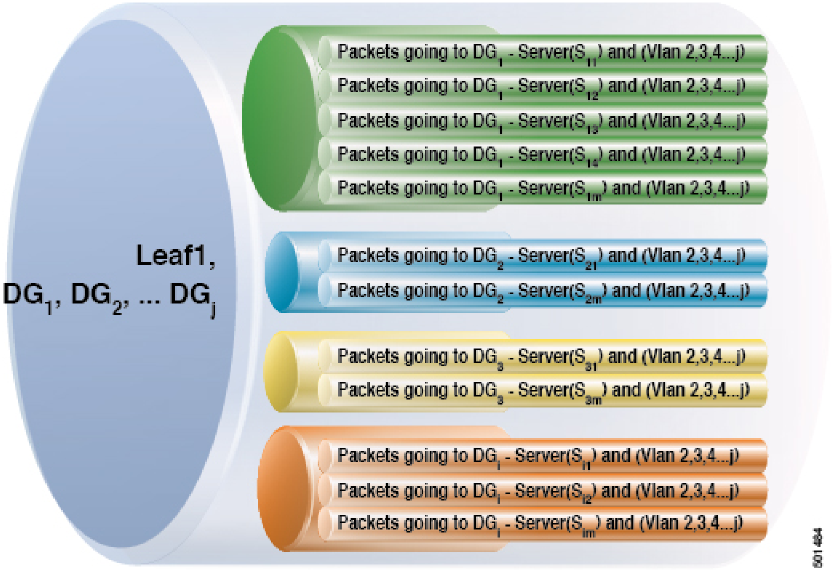

Leaf1,
DG_1, DG_2, ... DG_j
Packets going to DG_1 - Server(S_{11}) and (Vlan 2,3,4...j)
Packets going to DG_1 - Server(S_{12}) and (Vlan 2,3,4...j)
Packets going to DG_1 - Server(S_{13}) and (Vlan 2,3,4...j)
Packets going to DG_1 - Server(S_{14}) and (Vlan 2,3,4...j)
Packets going to DG_1 - Server(S_{1m}) and (Vlan 2,3,4...j)
Packets going to DG_2 - Server(S_{21}) and (Vlan 2,3,4...j)
Packets going to DG_2 - Server(S_{2m}) and (Vlan 2,3,4...j)
Packets going to DG_3 - Server(S_{31}) and (Vlan 2,3,4...j)
Packets going to DG_3 - Server(S_{3m}) and (Vlan 2,3,4...j)
Packets going to DG_j - Server(S_{j1}) and (Vlan 2,3,4...j)
Packets going to DG_j - Server(S_{j2}) and (Vlan 2,3,4...j)
Packets going to DG_j - Server(S_{jm}) and (Vlan 2,3,4...j)
501484

Figure 26: Analytics for Various Applications using Different Device Groups

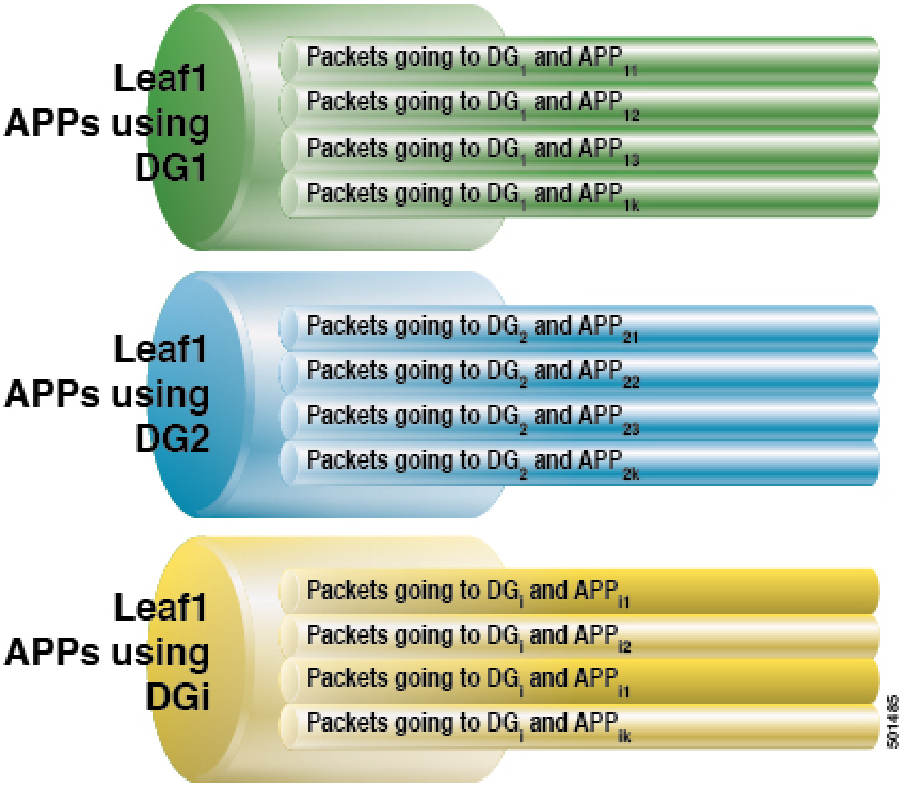
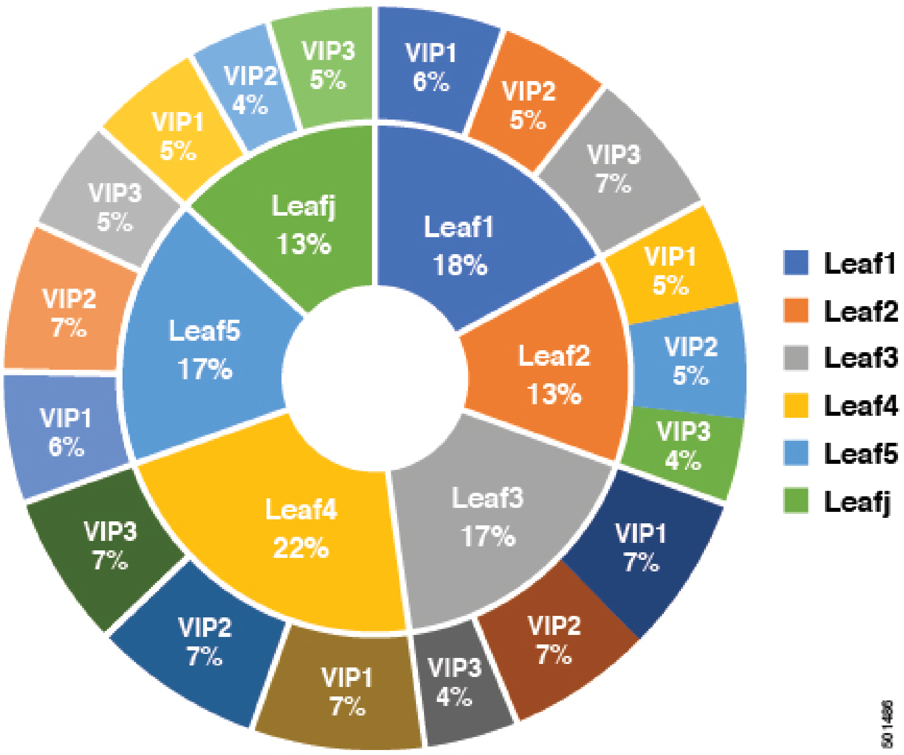

Leaf1
APPs using
DG1
Packets going to DG_1 and APP_{11}
Packets going to DG_1 and APP_{12}
Packets going to DG_1 and APP_{13}
Packets going to DG_1 and APP_{1k}
Leaf1
APPs using
DG2
Packets going to DG_2 and APP_{21}
Packets going to DG_2 and APP_{22}
Packets going to DG_2 and APP_{23}
Packets going to DG_2 and APP_{2k}
Leaf1
APPs using
DGi
Packets going to DG_i and APP_{i1}
Packets going to DG_i and APP_{i2}
Packets going to DG_i and APP_{i1}
Packets going to DG_i and APP_{ik}
501485

Figure 27: Apps (VIP) Traffic Analytics across Fabric



Verifying PLB Configuration

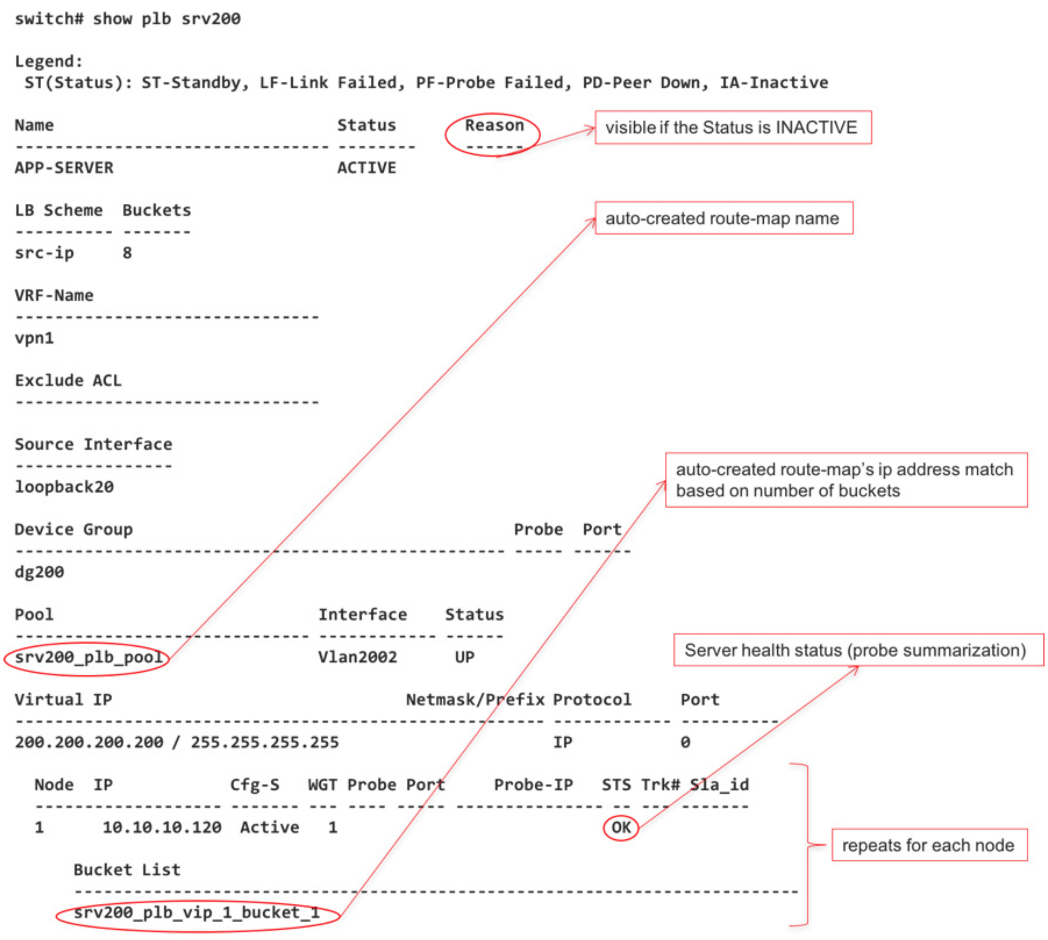
Use the following show commands to verify PLB configuration:

- **show running-config plb-services**—Displays the running configuration of all PLB services on a VDC or ToR/leaf switch.
- **show tech-support plb** [**detail**]—Displays the technical support information for PLB.
- **show plb** <*svc-name*> [**brief**]—Displays the current state of the specified PLB service or all services.
- **show plb** <*vrf*> <*svc-name*>—Displays the configured VRF of the specified PLB service or all services.
- **show plb analytics** <*svc-name*> [**brief**] —Displays the buckets, servers, VIP, and PLB service loads analytics for the specified PLB service or all services.

PLB Active Service Sample Outputs

The following shows an active PLB sample outputs:

Figure 28: Active Service Sample



```
switch# show plb srv200

Legend:
 ST(Status): ST-Standby, LF-Link Failed, PF-Probe Failed, PD-Peer Down, IA-Inactive

Name                             Status     Reason
-------------------------------- --------   ------
APP-SERVER                       ACTIVE

LB Scheme  Buckets
---------- -------
src-ip     8

VRF-Name
-------------------------------
vpn1

Exclude ACL
-------------------------------

Source Interface
----------------
loopback20

Device Group                                         Probe  Port
---------------------------------------------------- ----- ------
dg200

Pool                           Interface    Status
------------------------------ ------------ ------
srv200_plb_pool                Vlan2002      UP

Virtual IP                              Netmask/Prefix Protocol     Port
------------------------------------------------------ ----------- ----------
200.200.200.200 / 255.255.255.255                      IP           0

  Node  IP              Cfg-S   WGT Probe Port     Probe-IP    STS Trk# Sla_id
  ------------------ ------- --- ---- ----- --------------- -- --- -------
  1      10.10.10.120  Active   1                               OK

      Bucket List
      -------------------------------------------------------------------------
      srv200_plb_vip_1_bucket_1
```

visible if the Status is INACTIVE

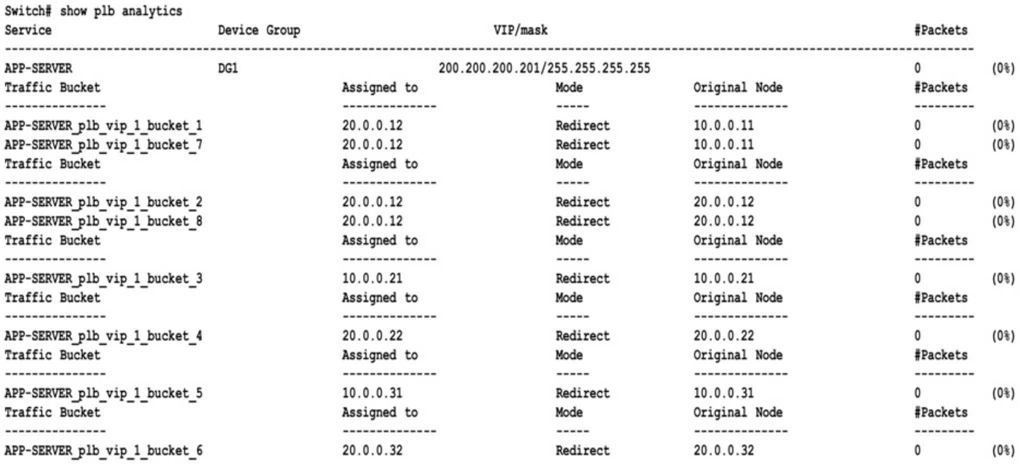
auto-created route-map name

auto-created route-map's ip address match based on number of buckets

Server health status (probe summarization)

repeats for each node

Figure 29: PLB Analytics Sample Output



```
Switch# show plb analytics
Service                   Device Group                          VIP/mask                                       #Packets
---------------------------------------------------------------------------------------------------------------------------
APP-SERVER                DG1                    200.200.200.201/255.255.255.255                               0          (0%)
Traffic Bucket                              Assigned to              Mode             Original Node            #Packets
---------------                             --------------           -----            --------------           ---------
APP-SERVER_plb_vip_1_bucket_1               20.0.0.12                Redirect         10.0.0.11                0          (0%)
APP-SERVER_plb_vip_1_bucket_7               20.0.0.12                Redirect         10.0.0.11                0          (0%)
Traffic Bucket                              Assigned to              Mode             Original Node            #Packets
---------------                             --------------           -----            --------------           ---------
APP-SERVER_plb_vip_1_bucket_2               20.0.0.12                Redirect         20.0.0.12                0          (0%)
APP-SERVER_plb_vip_1_bucket_8               20.0.0.12                Redirect         20.0.0.12                0          (0%)
Traffic Bucket                              Assigned to              Mode             Original Node            #Packets
---------------                             --------------           -----            --------------           ---------
APP-SERVER_plb_vip_1_bucket_3               10.0.0.21                Redirect         10.0.0.21                0          (0%)
Traffic Bucket                              Assigned to              Mode             Original Node            #Packets
---------------                             --------------           -----            --------------           ---------
APP-SERVER_plb_vip_1_bucket_4               20.0.0.22                Redirect         20.0.0.22                0          (0%)
Traffic Bucket                              Assigned to              Mode             Original Node            #Packets
---------------                             --------------           -----            --------------           ---------
APP-SERVER_plb_vip_1_bucket_5               10.0.0.31                Redirect         10.0.0.31                0          (0%)
Traffic Bucket                              Assigned to              Mode             Original Node            #Packets
---------------                             --------------           -----            --------------           ---------
APP-SERVER_plb_vip_1_bucket_6               20.0.0.32                Redirect         20.0.0.32                0          (0%)
```

CHAPTER 5

IP Fabric Underlay

IP Fabric Underlay

Underlay Considerations

Unicast underlay:

The primary purpose of the underlay in the VXLAN EVPN fabric is to advertise the reachability of Virtual Tunnel End Points (VTEPs) and BGP peering addresses. The primary criterion for choosing an underlay protocol is fast convergence in the event of node failures. Other criteria are:

- Simplicity of configuration.
- Ability to delay the introduction of a node into the network on boot up.

This document will detail the two primary protocols supported and tested by Cisco, IS-IS and OSPF. It will also illustrate the use of eBGP protocol as an underlay for the VXLAN EVPN fabric.

From an underlay/overlay perspective, the packet flow from a server to another over the Virtual Extensible LAN (VXLAN) fabric comprises of these steps:

1. Server sends traffic to source VXLAN tunnel endpoint (VTEP). The VTEP performs Layer-2 or Layer-3 communication based on the destination MAC and derives the nexthop (destination VTEP).

Note When a packet is bridged, the target end host's MAC address is stamped in the DMAC field of the inner frame. When a packet is routed, the default gateway's MAC address is stamped in the DMAC field of the inner frame.

2. The VTEP encapsulates the traffic (frames) into VXLAN packets (overlay function – see Figure 1) and signals the underlay IP network.
3. Based on the underlay routing protocol, the packet is sent from the source VTEP to destination VTEP through the IP network (underlay function – see *Underlay Overview* figure).

4. The destination VTEP removes the VXLAN encapsulation (overlay function) and sends traffic to the intended server.

The VTEPs are a part of the underlay network as well since VTEPs need to be reachable to each other to send VXLAN encapsulated traffic across the IP underlay network.

The *Overlay Overview* and *Underlay Overview* images (below) depict the broad difference between an overlay and underlay. Since the focus is on the VTEPs, the spine switches are only depicted in the background. Note that, in real time, the packet flow from VTEP to VTEP traverses through the spine switches.

Figure 30: Overlay Overview

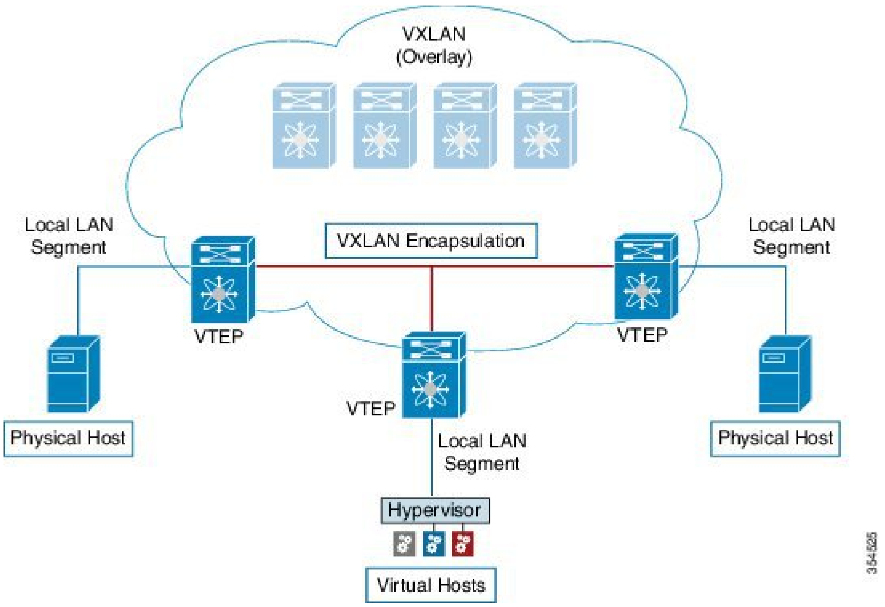
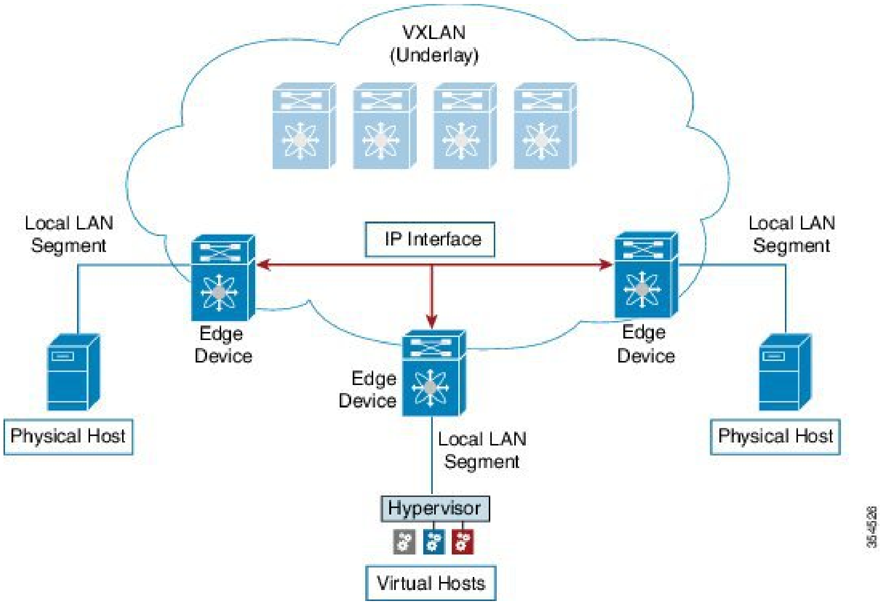


Figure 31: Underlay Overview



Deployment considerations for an underlay IP network in a VXLAN EVPN Programmable Fabric

The deployment considerations for an underlay IP network in a VXLAN EVPN Programmable Fabric are given below:

- Maximum transmission unit (MTU) – Due to VXLAN encapsulation, the MTU requirement is larger and we need to avoid potential fragmentation.
    - An MTU of 9216 bytes on each interface on the path between the VTEPs accommodates maximum server MTU + VXLAN overhead. Most datacenter server NICs support up to 9000 bytes. So, no fragmentation is needed for VXLAN traffic.

      Cisco Nexus 5600 series switches use a 24 byte internal header for switching packets between ASICs, reducing the MTU size of the interface to 9192.

      Note If the fabric only contains Cisco Nexus 9000 and 7000 series switches, then the MTU should be set to 9216.

    - The VXLAN IP fabric underlay supports IPv4 address family.

- Unicast routing - Any unicast routing protocol can be used for the VXLAN IP underlay. You can implement OSPF, IS-IS, or eBGP to route between the VTEPs.

  Note As a best practice, use a simple IGP (OSPF or IS-IS) for underlay reachability between VTEPs with iBGP for overlay information exchange.

- IP addressing – Point-to-point (P2P) or IP unnumbered links. For each point-to-point link, as example between the leaf switch nodes and spine switch nodes, typically a /30 IP mask should be assigned. Optionally a /31 mask or IP unnumbered links can be assigned. The IP unnumbered approach is leaner from an addressing perspective and consumes fewer IP addresses. The IP unnumbered option for the OSPF or IS-IS protocol underlay will minimize the use of IP addresses.

  /31 network - An OSPF or IS-IS point-to-point numbered network is only between two switch (interfaces), and there is no need for a broadcast or network address. So, a /31 network will suffice for this network. Neighbors on this network establish adjacency and there is no designated router (DR) for the network.

  Note IP Unnumbered for VXLAN underlay is supported starting with Cisco NX-OS Release 7.0(3)I7(2).

- Multicast protocol for multi destination (BUM) traffic – Though VXLAN has the BGP EVPN control plane, the VXLAN fabric still requires a technology for Broadcast/Unknown unicast/Multicast (BUM) traffic to be forwarded. For Cisco Nexus 5600 Series switches and Cisco Nexus 7000/7700 Series switches, it is mandatory to implement a multicast protocol for BUM packet communication.

  While Cisco Nexus 5600 Series switches support Protocol Independent Multicast (PIM) Bidirectional shared trees (BiDiR), Cisco Nexus 7000/7700 Series switches (with F3 cards) support PIM Any Source Multicast (ASM) and PIM BiDir options.

- PIM BiDir is supported for Cisco Nexus 9300-EX and 9300-FX/FX2/FX3 platform switches.
- vPC configuration — This is documented in Chapter 3. For comprehensive information on vPCs, refer to the respective Cisco Nexus 5600, 7000, or 9000 Series vPC design/configuration guide.

Unicast routing and IP addressing options

Each unicast routing protocol option (OSPF, IS-IS, and eBGP) and sample configurations are given below. Use an option to suit your setup's requirements.

Important All routing configuration samples are from an IP underlay perspective and are not comprehensive. For complete configuration information including routing process, authentication, Bidirectional Forwarding Detection (BFD) information, and so on, refer to the respective routing configuration guide (for example, *Cisco Nexus 5600 Series NX-OS Unicast Routing Configuration Guide*, *Cisco Nexus 7000 Series NX-OS Unicast Routing Configuration Guide*, and *Cisco Nexus 9000 Series NX-OS Unicast Routing Configuration Guide*).

OSPF Underlay IP Network

Some considerations are given below:

- For IP addressing, use P2P links. Since only two switches are directly connected, you can avoid a Designated Router/Backup Designated Router (DR/BDR) election.
- Use the *point-to-point* network type option. It is ideal for routed interfaces or ports, and is optimal from a Link State Advertisements (LSA) perspective.
- Do not use the broadcast type network. It is suboptimal from an LSA database perspective (LSA type 1 – Router LSA and LSA type 2 – Network LSA) and necessitates a DR/BDR election, thereby creating an additional election and database overhead.

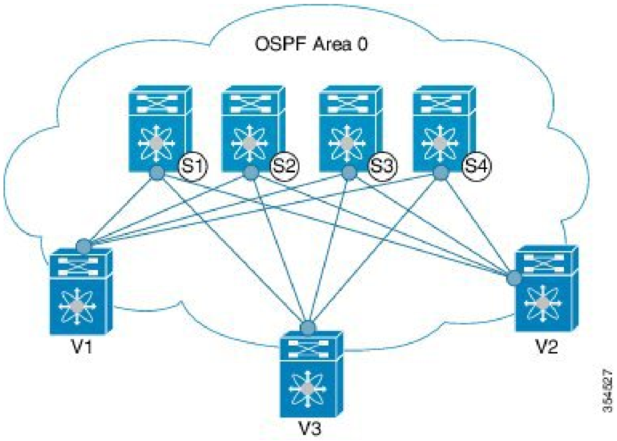
Note You can divide OSPF networks into areas when the size of the routing domain contains a high number of routers and/or IP prefixes.. The same general well known OSPF best practice rules in regards of scale and configuration are applicable for the VXLAN underlay too. For example, LSA type 1 and type 2 are never flooded outside of an area. With multiple areas, the size of the OSPF LSA databases can be reduced to optimize CPU and memory consumption.

Note

- For ease of use, the configuration mode from which you need to start configuring a task is mentioned at the beginning of each configuration.
- Configuration tasks and corresponding show command output are displayed for a part of the topology in the image. For example, if the sample configuration is shown for a leaf switch and connected spine switch, the show command output for the configuration displays corresponding configuration.

OSPF configuration sample – P2P and IP unnumbered network scenarios

Figure 32: OSPF as the underlay routing protocol



OSPF – P2P link scenario with /31 mask

In the above image, the leaf switches (V1, V2, and V3) are at the bottom of the image. They are connected to the 4 spine switches (S1, S2, S3, and S4) that are depicted at the top of the image. For P2P connections between a leaf switch (also having VTEP function) and each spine, leaf switches V1, V2, and V3 should each be connected to each spine switch.

For V1, we should configure a P2P interface to connect to each spine switch.

A sample P2P configuration between a leaf switch (V1) interface and a spine switch (S1) interface is given below:

OSPF global configuration on leaf switch V1

(config) #

```
feature ospf
router ospf UNDERLAY
 router-id 10.1.1.54
```

OSPF leaf switch V1 P2P interface configuration

(config) #

```
interface Ethernet 1/41
   description Link to Spine S1
   no switchport
   ip address 198.51.100.1/31
   mtu 9192
   ip router ospf UNDERLAY area 0.0.0.0
   ip ospf network point-to-point
```

The **ip ospf network point-to-point** command configures the OSPF network as a point-to-point network

The OSPF instance is tagged as UNDERLAY for better recall.

OSPF loopback interface configuration (leaf switch V1)

Configure a loopback interface so that it can be used as the OSPF router ID of leaf switch V1.

(config) #

```
interface loopback 0
   ip address 10.1.1.54/32
   ip router ospf UNDERLAY area 0.0.0.0
```

The interface will be associated with the OSPF instance UNDERLAY and OSPF area 0.0.0.0

OSPF global configuration on spine switch S1

(config) #

```
feature ospf
router ospf UNDERLAY
 router-id 10.1.1.53
```

(Corresponding) OSPF spine switch S1 P2P interface configuration

(config) #

```
interface Ethernet 1/41
   description Link to VTEP V1
   ip address 198.51.100.2/31
   mtu 9192
   ip router ospf UNDERLAY area 0.0.0.0
   ip ospf network point-to-point
   no shutdown
```

Use an MTU of 9192 for Cisco Nexus 5600 series switches.

Note MTU size of both ends of the link should be configured identically.

OSPF loopback Interface Configuration (spine switch S1)

Configure a loopback interface so that it can be used as the OSPF router ID of spine switch S1.

(config) #

```
interface loopback 0
   ip address 10.1.1.53/32
   ip router ospf UNDERLAY area 0.0.0.0
```

The interface will be associated with the OSPF instance UNDERLAY and OSPF area 0.0.0.0

.

.

To complete OSPF topology configuration for the ‘OSPF as the underlay routing protocol’ image, configure the following

- *3 more V1 interfaces (or 3 more P2P links) to the remaining 3 spine switches.*
- *Repeat the procedure to connect P2P links between V2,V3 and V4 and the spine switches.*

OSPF - IP unnumbered scenario

A sample OSPF IP unnumbered configuration is given below:

OSPF leaf switch V1 configuration

OSPF global configuration on leaf switch V1

(config) #

```
feature ospf
router ospf UNDERLAY
  router-id 10.1.1.54
```

The OSPF instance is tagged as UNDERLAY for better recall.

OSPF leaf switch V1 P2P interface configuration

(config) #

```
interface Ethernet1/41
  description Link to Spine S1
  mtu 9192
  ip ospf network point-to-point
  ip unnumbered loopback0
  ip router ospf UNDERLAY area 0.0.0.0
```

Use an MTU of 9192 for Cisco Nexus 5600 series switches.

The **ip ospf network point-to-point** command configures the OSPF network as a point-to-point network.

OSPF loopback interface configuration

Configure a loopback interface so that it can be used as the OSPF router ID of leaf switch V1.

(config) #

```
interface loopback0
  ip address 10.1.1.54/32
  ip router ospf UNDERLAY area 0.0.0.0
```

The interface will be associated with the OSPF instance UNDERLAY and OSPF area 0.0.0.0

OSPF spine switch S1 configuration:

OSPF global configuration on spine switch S1

(config) #

```
feature ospf
router ospf UNDERLAY
  router-id 10.1.1.53
```

(Corresponding) OSPF spine switch S1 P2P interface configuration

(config) #

```
interface Ethernet1/41
  description Link to VTEP V1
  mtu 9192
  ip ospf network point-to-point
  ip unnumbered loopback0
  ip router ospf UNDERLAY area 0.0.0.0
```

Use an MTU of 9192 for Cisco Nexus 5600 series switches.

OSPF loopback interface configuration (spine switch S1)

Configure a loopback interface so that it can be used as the OSPF router ID of spine switch S1.

(config) #

```
interface loopback0
  ip address 10.1.1.53/32
  ip router ospf UNDERLAY area 0.0.0.0
```

The interface will be associated with the OSPF instance UNDERLAY and OSPF area 0.0.0.0

.

.

To complete OSPF topology configuration for the ‘OSPF as the underlay routing protocol’ image, configure the following:

- *3 more VTEP V1 interfaces (or 3 more IP unnumbered links) to the remaining 3 spine switches.*
- *Repeat the procedure to connect IP unnumbered links between VTEPs V2,V3 and V4 and the spine switches.*

OSPF Verification

Use the following commands for verifying OSPF configuration:

```
Leaf-Switch-V1# show ip ospf

 Routing Process UNDERLAY with ID 10.1.1.54 VRF default
 Routing Process Instance Number 1
 Stateful High Availability enabled
 Graceful-restart is configured
   Grace period: 60 state: Inactive
   Last graceful restart exit status: None
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 Administrative distance 110
 Reference Bandwidth is 40000 Mbps
 SPF throttling delay time of 200.000 msecs,
   SPF throttling hold time of 1000.000 msecs,
   SPF throttling maximum wait time of 5000.000 msecs
 LSA throttling start time of 0.000 msecs,
   LSA throttling hold interval of 5000.000 msecs,
   LSA throttling maximum wait time of 5000.000 msecs
 Minimum LSA arrival 1000.000 msec
 LSA group pacing timer 10 secs
 Maximum paths to destination 8
 Number of external LSAs 0, checksum sum 0
 Number of opaque AS LSAs 0, checksum sum 0
 Number of areas is 1, 1 normal, 0 stub, 0 nssa
 Number of active areas is 1, 1 normal, 0 stub, 0 nssa
 Install discard route for summarized external routes.
 Install discard route for summarized internal routes.
   Area BACKBONE(0.0.0.0)
        Area has existed for 03:12:54
        Interfaces in this area: 2 Active interfaces: 2
        Passive interfaces: 0  Loopback interfaces: 1
        No authentication available
        SPF calculation has run 5 times
         Last SPF ran for 0.000195s
        Area ranges are
        Number of LSAs: 3, checksum sum 0x196c2


Leaf-Switch-V1# show ip ospf interface

loopback0 is up, line protocol is up
    IP address 10.1.1.54/32
    Process ID UNDERLAY VRF default, area 0.0.0.0
    Enabled by interface configuration
    State LOOPBACK, Network type LOOPBACK, cost 1
    Index 1
 Ethernet1/41 is up, line protocol is up
    Unnumbered interface using IP address of loopback0 (10.1.1.54)
    Process ID UNDERLAY VRF default, area 0.0.0.0
    Enabled by interface configuration
    State P2P, Network type P2P, cost 4
    Index 2, 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


Leaf-Switch-V1# show ip ospf neighbors

OSPF Process ID UNDERLAY VRF default
 Total number of neighbors: 1
 Neighbor ID     Pri State            Up Time  Address         Interface
 10.1.1.53         1 FULL/ -          06:18:32 10.1.1.53        Eth1/41
```

For a detailed list of commands, refer to the Configuration and Command Reference guides.

IS-IS Underlay IP Network

Some considerations are given below:

- Since IS-IS uses Connectionless Network Service (CLNS) and is independent of the IP, full SPF calculation is avoided when a link changes.
- Net ID - Each IS-IS instance has an associated network entity title (NET) ID that uniquely identifies the IS-IS instance in the area. The NET ID is comprised of the IS-IS system ID, which uniquely identifies this IS-IS instance in the area, and the area ID. For example, if the NET ID is 49.0001.0010.0100.1074.00, the system ID is 0010.0100.1074 and the area ID is 49.0001.

Important

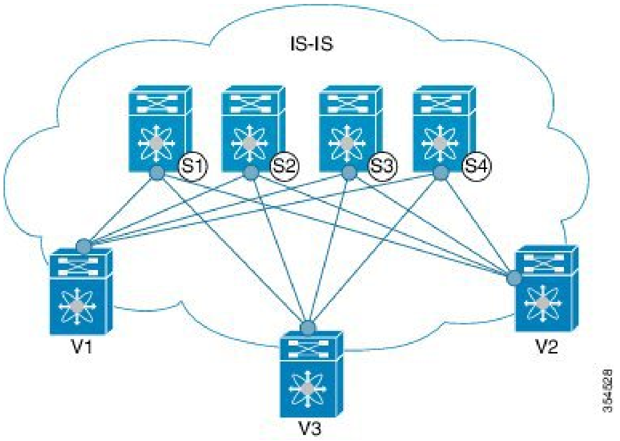
Level 1 IS-IS in the Fabric—Cisco has validated the use of IS-IS Level 1 only and IS-IS Level 2 only configuration on all nodes in the programmable fabric. The fabric is considered a stub network where every node needs an optimal path to every other node in the fabric. Cisco NX-OS IS-IS implementation scales well to support a number of nodes in a fabric, hence there is no anticipation of having to break up the fabric into multiple IS-IS domains.

Note

- For ease of use, the configuration mode from which you need to start configuring a task is mentioned at the beginning of each configuration.
- Configuration tasks and corresponding show command output are displayed for a part of the topology in the image. For example, if the sample configuration is shown for a leaf switch and connected spine switch, the show command output for the configuration displays corresponding configuration.

IS-IS configuration sample - P2P and IP unnumbered network scenarios

Figure 33: IS-IS as the underlay routing protocol



In the above image, the leaf switches (V1, V2, and V3, having the VTEP function) are at the bottom of the image. They are connected to the 4 spine switches (S1, S2, S3, and S4) that are depicted at the top of the image.

IS-IS – P2P link scenario with /31 mask

A sample P2P configuration between V1 and spine switch S1 is given below:

For P2P connections between a leaf switch and each spine switch, V1, V2, and V3 should each be connected to each spine switch.

For V1, we must configure a loopback interface and a P2P interface configuration to connect to S1. A sample P2P configuration between a leaf switch (V1) interface and a spine switch (S1) interface is given below:

IS-IS configuration on leaf switch V1

IS-IS global configuration

(config) #

```
feature isis
router isis UNDERLAY
  net 49.0001.0010.0100.1074.00
  is-type level-1
  set-overload-bit on-startup 60
```

Setting the overload bit - You can configure a Cisco Nexus switch to signal other devices not to use the switch as an intermediate hop in their shortest path first (SPF) calculations. You can optionally configure the overload bit temporarily on startup. In the above example, the **set-overload-bit** command is used to set the overload bit on startup to 60 seconds.

IS-IS P2P interface configuration (leaf switch V1)

(config) #

```
interface Ethernet 1/41
   description Link to Spine S1
   mtu 9192
   ip address 209.165.201.1/31
   ip router isis UNDERLAY
```

Use an MTU of 9192 for Cisco Nexus 5600 series switches.

IS-IS loopback interface configuration (leaf switch V1)

Configure a loopback interface so that it can be used as the IS-IS router ID of leaf switch V1.

(config) #

```
interface loopback 0
   ip address 10.1.1.74/32
   ip router isis UNDERLAY
```

The IS-IS instance is tagged as UNDERLAY for better recall.

(Corresponding) IS-IS spine switch S1 configuration

IS-IS global configuration

(config) #

```
feature isis
router isis UNDERLAY
 net 49.0001.0010.0100.1053.00
 is-type level-1
 set-overload-bit on-startup 60
```

IS-IS P2P interface configuration (spine switch S1)

(config) #

```
interface Ethernet 1/1
   description Link to VTEP V1
   ip address 209.165.201.2/31
   mtu 9192
   ip router isis UNDERLAY
```

Use an MTU of 9192 for Cisco Nexus 5600 series switches.

IS-IS loopback interface configuration (spine switch S1)

(config) #

```
interface loopback 0
   ip address 10.1.1.53/32
   ip router isis UNDERLAY
.
.
```

To complete IS-IS topology configuration for the above image, configure the following:

- *3 more leaf switch V1's interfaces (or 3 more P2P links) to the remaining 3 spine switches.*
- *Repeat the procedure to connect P2P links between leaf switches V2, V3 and V4 and the spine switches.*

IS-IS - IP unnumbered scenario

IS-IS configuration on leaf switch V1

IS-IS global configuration

(config)#

```
feature isis
router isis UNDERLAY
  net 49.0001.0010.0100.1074.00
  is-type level-1
  set-overload-bit on-startup 60
```

IS-IS interface configuration (leaf switch V1)

(config) #

```
interface Ethernet1/41
  description Link to Spine S1
  mtu 9192
  medium p2p
  ip unnumbered loopback0
  ip router isis UNDERLAY
```

Use an MTU of 9192 for Cisco Nexus 5600 series switches.

IS-IS loopback interface configuration (leaf switch V1)

(config)

```
interface loopback0
  ip address 10.1.1.74/32
  ip router isis UNDERLAY
```

IS-IS configuration on the spine switch S1

IS-IS global configuration

(config)#

```
feature isis
router isis UNDERLAY
  net 49.0001.0010.0100.1053.00
  is-type level-1
  set-overload-bit on-startup 60
```

IS-IS interface configuration (spine switch S1)

(config)#

```
interface Ethernet1/41
  description Link to V1
  mtu 9192
  medium p2p
  ip unnumbered loopback0
  ip router isis UNDERLAY
```

Use an MTU of 9192 for Cisco Nexus 5600 series switches.

IS-IS loopback interface configuration (spine switch S1)

(config)#

```
interface loopback0
  ip address 10.1.1.53/32
  ip router isis UNDERLAY
```

IS-IS Verification

Use the following commands for verifying IS-IS configuration on leaf switch V1:

```
Leaf-Switch-V1# show isis

ISIS process : UNDERLAY
 Instance number :  1
 UUID: 1090519320
 Process ID 20258
VRF: default
  System ID : 0010.0100.1074  IS-Type : L1
  SAP : 412  Queue Handle : 15
  Maximum LSP MTU: 1492
  Stateful HA enabled
  Graceful Restart enabled. State: Inactive
  Last graceful restart status : none
  Start-Mode Complete
  BFD IPv4 is globally disabled for ISIS process: UNDERLAY
  BFD IPv6 is globally disabled for ISIS process: UNDERLAY
  Topology-mode is base
  Metric-style : advertise(wide), accept(narrow, wide)
  Area address(es) :
    49.0001
 Process is up and running
  VRF ID: 1
  Stale routes during non-graceful controlled restart
  Interfaces supported by IS-IS :
    loopback0
    loopback1
    Ethernet1/41
  Topology : 0
  Address family IPv4 unicast :
    Number of interface : 2
    Distance : 115
  Address family IPv6 unicast :
    Number of interface : 0
    Distance : 115
  Topology : 2
  Address family IPv4 unicast :
    Number of interface : 0
    Distance : 115
  Address family IPv6 unicast :
    Number of interface : 0
    Distance : 115
   Level1
  No auth type and keychain
  Auth check set
  Level2
  No auth type and keychain
  Auth check set
  L1 Next SPF: Inactive
  L2 Next SPF: Inactive


Leaf-Switch-V1# show isis interface

IS-IS process: UNDERLAY VRF: default
loopback0, Interface status: protocol-up/link-up/admin-up IP address: 10.1.1.74, IP subnet:
 10.1.1.74/32
IPv6 routing is disabled Level1
No auth type and keychain Auth check set
Level2
No auth type and keychain Auth check set
Index: 0x0001, Local Circuit ID: 0x01, Circuit Type: L1 BFD IPv4 is locally disabled for
Interface loopback0 BFD IPv6 is locally disabled for Interface loopback0 MTR is disabled
Level Metric 1  1
2 1
Topologies enabled:
    L  MT  Metric  MetricCfg  Fwdng IPV4-MT  IPV4Cfg  IPV6-MT  IPV6Cfg
    1  0        1       no   UP    UP       yes      DN       no
    2  0        1       no   DN    DN       no       DN       no

loopback1, Interface status: protocol-up/link-up/admin-up
  IP address: 10.1.2.74, IP subnet: 10.1.2.74/32
  IPv6 routing is disabled
  Level1
    No auth type and keychain
    Auth check set
  Level2
    No auth type and keychain
    Auth check set
  Index: 0x0002, Local Circuit ID: 0x01, Circuit Type: L1
  BFD IPv4 is locally disabled for Interface loopback1
  BFD IPv6 is locally disabled for Interface loopback1
  MTR is disabled
  Passive level: level-2
  Level      Metric
  1               1
  2               1
  Topologies enabled:
    L  MT  Metric  MetricCfg  Fwdng IPV4-MT  IPV4Cfg  IPV6-MT  IPV6Cfg
    1  0        1       no   UP    UP       yes      DN       no
    2  0        1       no   DN    DN       no       DN       no

Ethernet1/41, Interface status: protocol-up/link-up/admin-up
  IP unnumbered interface (loopback0)
  IPv6 routing is disabled
    No auth type and keychain
    Auth check set
  Index: 0x0002, Local Circuit ID: 0x01, Circuit Type: L1
  BFD IPv4 is locally disabled for Interface Ethernet1/41
  BFD IPv6 is locally disabled for Interface Ethernet1/41
  MTR is disabled
  Extended Local Circuit ID: 0x1A028000, P2P Circuit ID: 0000.0000.0000.00
  Retx interval: 5, Retx throttle interval: 66 ms
  LSP interval: 33 ms, MTU: 9192
  P2P Adjs: 1, AdjsUp: 1, Priority 64
  Hello Interval: 10, Multi: 3, Next IIH: 00:00:01
  MT    Adjs   AdjsUp  Metric   CSNP  Next CSNP  Last LSP ID
  1         1        1       4     60  00:00:35   ffff.ffff.ffff.ff-ff
  2         0        0       4     60  Inactive   ffff.ffff.ffff.ff-ff
  Topologies enabled:
    L  MT  Metric  MetricCfg  Fwdng IPV4-MT  IPV4Cfg  IPV6-MT  IPV6Cfg
    1  0        4       no   UP    UP       yes      DN       no
    2  0        4       no   UP    DN       no       DN       no


Leaf-Switch-V1# show isis adjacency

IS-IS process: UNDERLAY VRF: default
IS-IS adjacency database:
Legend: '!': No AF level connectivity in given topology
System ID          SNPA    Level  State Hold Time  Interface
Spine-Switch-S1    N/A     1      UP    00:00:23   Ethernet1/41
```

For a detailed list of commands, refer to the Configuration and Command Reference guides.

eBGP Underlay IP Network

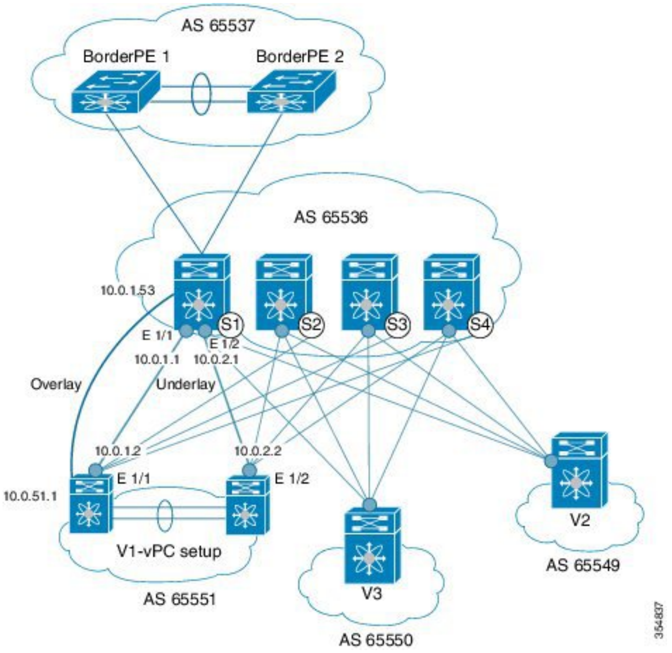
Some customers would like to have the same protocol in the underlay and overlay in order to contain the number of protocols that need support in their network.

There are various ways to configure the eBGP based underlay. The configurations given in this section have been validated for function and convergence. The IP underlay based on eBGP can be built with these configurations detailed below. (For reference, see image below)

- The design below is following the multi AS model.
- eBGP underlay requires numbered interfaces between leaf and spine nodes. Numbered interfaces are used for the underlay BGP sessions as there is no other protocol to distribute peer reachability.
- The overlay sessions are configured on loopback addresses. This is to increase the resiliency in presence of link or node failures.
- BGP speakers on spine layer configure all leaf node eBGP neighbors individually. This is different from IBGP based peering which can be covered by dynamic BGP.
- Pointers for Multiple AS numbers in a fabric are given below:
    - All spine nodes configured as BGP speakers are in one AS.
    - All leaf nodes will have a unique AS number that is different than the BGP speakers in spine layer.
    - A pair of vPC leaf switch nodes, have the same AS number.
    - If a globally unique AS number is required to represent the fabric, then that can be configured on the border leaf or borderPE switches. All other nodes can use the private AS number range.

- BGP Confederation has not been leveraged.

Figure 34: eBGP as underlay



eBGP configuration sample

Sample configurations for a spine switch and leaf switch are given below. The complete configuration is given for providing context, and the configurations added specifically for eBGP underlay are highlighted and further explained.

There is one BGP session per neighbor to set up the underlay. This is done within the global IPv4 address family. The session is used to distribute the loopback addresses for VTEP, Rendezvous Point (RP) and the eBGP peer address for the overlay eBGP session.

Spine switch S1 configuration—On the spine switch (S1 in this example), all leaf nodes are configured as eBGP neighbors.

(config) #

```
router bgp 65536
   router-id 10.1.1.53
   address-family ipv4 unicast
   redistribute direct route-map DIRECT-ROUTES-MAP
```

The **redistribute direct** command is used to advertise the loopback addresses for BGP and VTEP peering. It can be used to advertise any other direct routes in the global address space. The route map can filter the advertisement to include only eBGP peering and VTEP loopback addresses.

```
   maximum-paths 2
   address-family l2vpn evpn
     retain route-target all
```

Spine switch BGP speakers don’t have any VRF configuration. Hence, the **retain route-target all** command is needed to retain the routes and send them to leaf switch VTEPs. The **maximum-paths** command is used for ECMP path in the underlay.

Underlay session towards leaf switch V1 (vPC set up) —As mentioned above, the underlay sessions are configured on the numbered interfaces between spine and leaf switch nodes.

(config) #

```
neighbor 10.0.1.2 remote-as 65551
   address-family ipv4 unicast
      disable-peer-as-check
      send-community both
```

The vPC pair of switches has the same AS number. The **disable-peer-as-check** command is added to allow route propagation between the vPC switches as they are configured with the same AS, for example, for route type 5 routes. If the vPC switches have different AS numbers, this command is not required.

Underlay session towards the border leaf switch—The underlay configurations towards leaf and border leaf switches are the same, barring the changes in IP address and AS values.

Overlay session on the spine switch S1 towards the leaf switch V1

(config) #

```
route-map UNCHANGED permit 10
   set ip next-hop unchanged
```

Note The route-map UNCHANGED is user defined whereas the keyword **unchanged** is an option within the **set ip next-hop** command. In eBGP, the next hop is changed to self when sending a route from one eBGP neighbor to another. The route map UNCHANGED is added to make sure that, for overlay routes, the originating leaf switch is set as next hop and not the spine switch. This ensures that VTEPs are next hops, and not spine switch nodes. The **unchanged** keyword ensures that the next-hop attribute in the BGP update to the eBGP peer is unmodified.

The overlay sessions are configured on loopback addresses.

(config) #

```
neighbor 10.0.51.1 remote-as 65551
   update-source loopback0
   ebgp-multihop 2
   address-family l2vpn evpn
     rewrite-evpn-rt-asn
       disable-peer-as-check
       send-community both
       route-map UNCHANGED out
```

The spine switch configuration concludes here. The *Route Target auto* feature configuration is given below for reference purposes:

(config) #

```
vrf context coke
    vni 50000
    rd auto
    address-family ipv4 unicast
      route-target both auto
      route-target both auto evpn
    address-family ipv6 unicast
      route-target both auto
      route-target both auto evpn
```

The **rewrite-evpn-rt-asn** command is required if the *Route Target auto* feature is being used to configure EVPN RTs.

Route target auto is derived from the Local AS number configured on the switch and the Layer-3 VNID of the VRF i.e. Local AS:VNID. In Multi-AS topology, as illustrated in this guide, each leaf node is represented as a different local AS, and the route target generated for the same VRF will be different on every switch. The command **rewrite-evpn-rt-asn** replaces the ASN portion of the route target in the BGP update message with the local AS number. For example, if VTEP V1 has a Local AS 65551, VTEP V2 has a Local AS 65549, and spine switch S1 has a Local AS 65536, then the route targets for V1, V2 and S1 are as follows:

- V1—65551:50000
- V2—65549:50000
- S1—65536:50000

In this scenario, V2 advertises the route with RT 65549:50000, the spine switch S1 replaces it with RT 65536:50000, and finally when V1 gets the update, it replaces the route target in the update with 65551:50000. This matches the locally configured RT on V1. This command requires that it be configured on all BGP speakers in the fabric.

If the *Route Target auto* feature is not being used, i.e., matching RTs are required to be manually configured on all switches, then this command is not necessary.

Leaf switch VTEP V1 configuration—In the sample configuration below, VTEP V1's interfaces are designated as BGP neighbors. All leaf switch VTEPs including border leaf switch nodes have the following configurations towards spine switch neighbor nodes:

(config) #

```
router bgp 65551
  router-id 10.1.1.54
  address-family ipv4 unicast
     maximum-paths 2
     address-family l2vpn evpn
```

The **maximum-paths** command is used for ECMP path in the underlay.

Underlay session on leaf switch VTEP V1 towards spine switch S1

(config) #

```
neighbor 10.0.1.1 remote-as 65536
   address-family ipv4 unicast
     allowas-in
     send-community both
```

The **allowas-in** command is needed if leaf switch nodes have the same AS. In particular, the Cisco validated topology had a vPC pair of switches share an AS number.

Overlay session towards spine switch S1

(config) #

```
neighbor 10.1.1.53 remote-as 65536
   update-source loopback0
   ebgp-multihop 2
   address-family l2vpn evpn
   rewrite-evpn-rt-asn
     allowas-in
     send-community both
```

The **ebgp-multihop 2** command is needed as the peering for the overlay is on the loopback address. NX-OS considers that as multi hop even if the neighbor is one hop away.

vPC backup session

(config) #

```
route-map SET-PEER-AS-NEXTHOP permit 10
  set ip next-hop peer-address

neighbor 192.168.0.1 remote-as 65551
    update-source Vlan3801
    address-family ipv4 unicast
      send-community both
      route-map SET-PEER-AS-NEXTHOP out
```

Note This session is configured on the backup SVI between the vPC leaf switch nodes.

To complete configurations for the above image, configure the following:

- *V1 as a BGP neighbor to other spine switches.*
- *Repeat the procedure for other leaf switches.*

BGP Verification

Use the following commands for verifying BGP configuration:

```
 show bgp all
 show bgp ipv4 unicast neighbors
 show ip route bgp
```

For a detailed list of commands, refer to the Configuration and Command Reference guides.

Multicast Routing in the VXLAN Underlay

The VXLAN EVPN Programmable Fabric supports multicast routing for transporting BUM (broadcast, unknown unicast and multicast) traffic.

Refer the table below to know the multicast protocol(s) your Cisco Nexus switches support:

Cisco Nexus Series Switch(es) Combination	Multicast Routing Option
Cisco Nexus 7000/7700 Series switches with Cisco Nexus 5600 Series switches	PIM BiDir
Cisco Nexus 7000/7700 Series switches with Cisco Nexus 9000 Series switches	PIM ASM (Sparse Mode)
Cisco Nexus 9000 Series	PIM ASM (Sparse Mode) *or* PIM BiDir **Note** PIM BiDir is supported on Cisco Nexus 9300-EX and 9300-FX/FX2/FX3 platform switches. PIM BiDir is not supported on Cisco Nexus 9300-GX platform switches.
Cisco Nexus 7000/7700 Series switches	PIM ASM (Sparse Mode) *or* PIM BiDir
Cisco Nexus 5600 Series switches	PIM BiDir

Note For Cisco Nexus 7000/7700 Series switches, an F3 or M3 card is required to support Cisco Programmable Fabric.

You can transport BUM traffic without multicast, through *ingress replication*. Ingress replication is currently available on Cisco Nexus 9000 Series switches.

PIM ASM and PIM BiDir Underlay IP Network

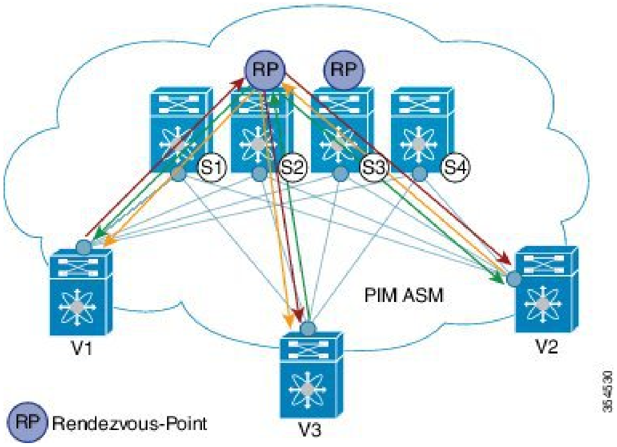
Some multicast topology design pointers are given below:

- Use spine/aggregation switches as Rendezvous-Point locations.
- Reserve a range of multicast groups (destination groups/DGroups) to service the overlay and optimize for diverse VNIs.
- In a spine-leaf topology with a lean spine,
  - Use multiple Rendezvous-Points across multiple spine switches.
  - Use redundant Rendezvous-Points.
  - Map different VNIs to different multicast groups, which are mapped to different Rendezvous-Points for load balancing.

Important The following configuration samples are from an IP underlay perspective and are not comprehensive. Functions such as PIM authentication, BFD for PIM, etc, are not shown here. Refer to the respective Cisco Nexus Series switch multicast configuration guide for complete information.

PIM Sparse-Mode (Any-Source Multicast [ASM])

Figure 35: PIM ASM as the IP multicast routing protocol



PIM ASM is supported on the Nexus 7000 and Nexus 9000 series as the underlay multicast protocol. (Nexus 7000 also supports bidirectional PIM as the underlay multicast protocol).

In the above image, the leaf switches (V1, V2, and V3 having VTEP configuration) are at the bottom of the image. They are connected to the 4 spine switches (S1, S2, S3, and S4) that are depicted at the top of the image.

Two multicast Rendezvous-Points (S2 and S3) are configured. The second Rendezvous-Point is added for load sharing and redundancy purposes. *Anycast RP is represented in the PIM ASM topology image*. Anycast RP ensures redundancy and load sharing between the two Rendezvous-Points. To use Anycast RP, multiple spines serving as RPs will share the same IP address (the Anycast RP address). Meanwhile, each RP has its unique IP address added in the RP set for RPs to sync information with respect to sources between all spines which act as RPs.

The shared multicast tree is unidirectional, and uses the Rendezvous-Point for forwarding packets.

PIM ASM at a glance - 1 source tree per multicast group per leaf switch.

Programmable Fabric specific pointers are:

- All VTEPs that serve a VNI join a shared multicast tree. VTEPs V1, V2, and V3 have hosts attached from a single tenant (say x) and these VTEPs form a separate multicast (source, group) tree.
- A VTEP (say V1) might have hosts belonging to other tenants too. Each tenant may have different multicast groups associated with. A source tree is created for each tenant residing on the VTEP, if the tenants do not share a multicast group.

PIM ASM Configuration

The PIM ASM examples are for the Cisco Nexus 7000 and 9000 Series switches.

Note For ease of use, the configuration mode from which you need to start configuring a task is mentioned at the beginning of each configuration.

Configuration tasks and corresponding show command output are displayed for a part of the topology in the image. For example, if the sample configuration is shown for a leaf switch and connected spine switch, the show command output for the configuration only displays corresponding configuration.

Leaf switch V1 Configuration — Configure RP reachability on the leaf switch.

PIM Anycast Rendezvous-Point association on leaf switch V1

(config) #

```
feature pim
ip pim rp-address 198.51.100.220 group-list 224.1.1.1
```

198.51.100.220 is the Anycast Rendezvous-Point IP address.

Loopback interface PIM configuration on leaf switch V1

(config) #

```
interface loopback 0
  ip address 209.165.201.20/32
  ip pim sparse-mode
```

Point-2-Point (P2P) interface PIM configuration for leaf switch V1 to spine switch S2 connectivity

(config) #

```
interface Ethernet 1/1
  no switchport
  ip address 209.165.201.14/31
  mtu 9216
  ip pim sparse-mode
.
.
```

Repeat the above configuration for a P2P link between V1 and the spine switch (S3) acting as the redundant Anycast Rendezvous-Point.

The VTEP also needs to be connected with spine switches (S1 and S4) that are not rendezvous points. A sample configuration is given below:

Point-2-Point (P2P) interface configuration for leaf switch V1 to non-rendezvous point spine switch (S1) connectivity

(config) #

```
interface Ethernet 2/2
  no switchport
  ip address 209.165.201.10/31
  mtu 9216
  ip pim sparse-mode
```

Repeat the above configuration for all P2P links between V1 and non- rendezvous point spine switches.

Repeat the complete procedure given above to configure all other leaf switches.

Rendezvous Point Configuration on the spine switches

PIM configuration on spine switch S2

(config) #

```
feature pim
```

Loopback Interface Configuration (RP)

(config) #

```
interface loopback 0
 ip address 10.10.100.100/32
 ip pim sparse-mode
```

Loopback interface configuration (Anycast RP)

(config) #

```
interface loopback 1
  ip address 198.51.100.220/32
  ip pim sparse-mode
```

Anycast-RP configuration on spine switch S2

Configure a spine switch as a Rendezvous Point and associate it with the loopback IP addresses of switches S2 and S3 for redundancy.

(config) #

```
feature pim
ip pim rp-address 198.51.100.220 group-list 224.1.1.1
ip pim anycast-rp 198.51.100.220 10.10.100.100
ip pim anycast-rp 198.51.100.220 10.10.20.100
.
.
```

Note The above configurations should also be implemented on the other spine switch (S3) performing the role of RP.

Non-RP Spine Switch Configuration

You also need to configure PIM ASM on spine switches that are not designated as rendezvous points, namely S1 and S4.

Earlier, leaf switch (VTEP) V1 has been configured for a P2P link to a non RP spine switch. A sample configuration on the non RP spine switch is given below.

PIM ASM global configuration on spine switch S1 (non RP)

(config) #

```
feature pim
ip pim rp-address 198.51.100.220 group-list 224.1.1.1
```

Loopback interface configuration (non RP)

(config) #

```
interface loopback 0
  ip address 10.10.100.103/32
  ip pim sparse-mode
```

Point-2-Point (P2P) interface configuration for spine switch S1 to leaf switch V1 connectivity

(config) #

```
interface Ethernet 2/2
  no switchport
  ip address 209.165.201.15/31
  mtu 9216
  ip pim sparse-mode
.
.
```

Repeat the above configuration for all P2P links between the non- rendezvous point spine switches and other leaf switches (VTEPs).

PIM ASM Verification

Use the following commands for verifying PIM ASM configuration:

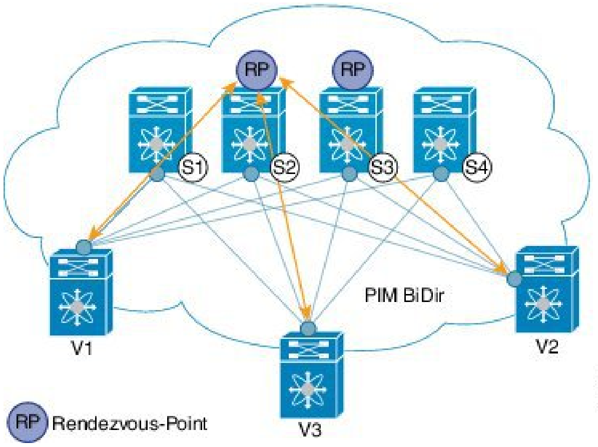
```
Leaf-Switch-V1# show ip mroute 224.1.1.1

IP Multicast Routing Table for VRF "default"

(*, 224.1.1.1/32), uptime: 02:21:20, nve ip pim
  Incoming interface: Ethernet1/1, RPF nbr: 10.10.100.100
  Outgoing interface list: (count: 1)
    nve1, uptime: 02:21:20, nve

(10.1.1.54/32, 224.1.1.1/32), uptime: 00:08:33, ip mrib pim
  Incoming interface: Ethernet1/2, RPF nbr: 209.165.201.12
  Outgoing interface list: (count: 1)
    nve1, uptime: 00:08:33, mrib

(10.1.1.74/32, 224.1.1.1/32), uptime: 02:21:20, nve mrib ip pim
  Incoming interface: loopback0, RPF nbr: 10.1.1.74
  Outgoing interface list: (count: 1)
    Ethernet1/6, uptime: 00:29:19, pim


Leaf-Switch-V1# show ip pim rp
```

```
PIM RP Status Information for VRF "default"
BSR disabled
Auto-RP disabled
BSR RP Candidate policy: None
BSR RP policy: None
Auto-RP Announce policy: None
Auto-RP Discovery policy: None

RP: 198.51.100.220, (0), uptime: 03:17:43, expires: never,
  priority: 0, RP-source: (local), group ranges:
      224.0.0.0/9


Leaf-Switch-V1# show ip pim interface

PIM Interface Status for VRF "default"
Ethernet1/1, Interface status: protocol-up/link-up/admin-up
  IP address: 209.165.201.14, IP subnet: 209.165.201.14/31
  PIM DR: 209.165.201.12, DR's priority: 1
  PIM neighbor count: 1
  PIM hello interval: 30 secs, next hello sent in: 00:00:11
  PIM neighbor holdtime: 105 secs
  PIM configured DR priority: 1
  PIM configured DR delay: 3 secs
  PIM border interface: no
  PIM GenID sent in Hellos: 0x33d53dc1
  PIM Hello MD5-AH Authentication: disabled
  PIM Neighbor policy: none configured
  PIM Join-Prune inbound policy: none configured
  PIM Join-Prune outbound policy: none configured
  PIM Join-Prune interval: 1 minutes
  PIM Join-Prune next sending: 1 minutes
  PIM BFD enabled: no
  PIM passive interface: no
  PIM VPC SVI: no
  PIM Auto Enabled: no
  PIM Interface Statistics, last reset: never
    General (sent/received):
      Hellos: 423/425 (early: 0), JPs: 37/32, Asserts: 0/0
      Grafts: 0/0, Graft-Acks: 0/0
      DF-Offers: 4/6, DF-Winners: 0/197, DF-Backoffs: 0/0, DF-Passes: 0/0
    Errors:
      Checksum errors: 0, Invalid packet types/DF subtypes: 0/0
      Authentication failed: 0
      Packet length errors: 0, Bad version packets: 0, Packets from self: 0
      Packets from non-neighbors: 0
          Packets received on passiveinterface: 0
      JPs received on RPF-interface: 0
      (*,G) Joins received with no/wrong RP: 0/0
      (*,G)/(S,G) JPs received for SSM/Bidir groups: 0/0
      JPs filtered by inbound policy: 0
      JPs filtered by outbound policy: 0
loopback0, Interface status: protocol-up/link-up/admin-up
  IP address: 209.165.201.20, IP subnet: 209.165.201.20/32
  PIM DR: 209.165.201.20, DR's priority: 1
  PIM neighbor count: 0
  PIM hello interval: 30 secs, next hello sent in: 00:00:07
  PIM neighbor holdtime: 105 secs
  PIM configured DR priority: 1
  PIM configured DR delay: 3 secs
  PIM border interface: no
  PIM GenID sent in Hellos: 0x1be2bd41
  PIM Hello MD5-AH Authentication: disabled
  PIM Neighbor policy: none configured
  PIM Join-Prune inbound policy: none configured
  PIM Join-Prune outbound policy: none configured
  PIM Join-Prune interval: 1 minutes
  PIM Join-Prune next sending: 1 minutes
  PIM BFD enabled: no
  PIM passive interface: no
  PIM VPC SVI: no
  PIM Auto Enabled: no
  PIM Interface Statistics, last reset: never
    General (sent/received):
      Hellos: 419/0 (early: 0), JPs: 2/0, Asserts: 0/0
      Grafts: 0/0, Graft-Acks: 0/0
      DF-Offers: 3/0, DF-Winners: 0/0, DF-Backoffs: 0/0, DF-Passes: 0/0
    Errors:
      Checksum errors: 0, Invalid packet types/DF subtypes: 0/0
      Authentication failed: 0
      Packet length errors: 0, Bad version packets: 0, Packets from self: 0
     Packets from non-neighbors: 0
          Packets received on passiveinterface: 0
      JPs received on RPF-interface: 0
      (*,G) Joins received with no/wrong RP: 0/0
      (*,G)/(S,G) JPs received for SSM/Bidir groups: 0/0
      JPs filtered by inbound policy: 0
      JPs filtered by outbound policy: 0


Leaf-Switch-V1# show ip pim neighbor

PIM Neighbor Status for VRF "default"

Neighbor       Interface      Uptime     Expires     DR         Bidir-        BFD
                                                     Priority   Capable       State
10.10.100.100  Ethernet1/1    1w1d       00:01:33    1          yes           n/a
```

For a detailed list of commands, refer to the Configuration and Command Reference guides.

PIM Bidirectional (BiDir)

Figure 36: PIM BiDir as the IP multicast routing protocol



Bidirectional PIM is supported on the Nexus 5600 and Nexus 7000 series as the underlay multicast protocol. Some multicast topology design pointers are given below:

VXLAN BiDir underlay is supported on Cisco Nexus 9300-EX and 9300-FX/FX2/FX3 platform switches.

In the above image, the leaf switches (V1, V2, and V3) are at the bottom of the image. They are connected to the 4 spine switches (S1, S2, S3, and S4) that are depicted at the top of the image. The two PIM Rendezvous-Points using phantom RP mechanism are used for load sharing and redundancy purposes.

Note Load sharing happens only via different multicast groups, for the respective, different VNI.

With bidirectional PIM, one bidirectional, shared tree rooted at the RP is built for each multicast group. Source specific state are not maintained within the fabric which provides a more scalable solution.

Programmable Fabric specific pointers are:

- The 3 VTEPs share the same VNI and multicast group mapping to form a single multicast group tree.

PIM BiDir at a glance — *One shared tree per multicast group*.

PIM BiDir Configuration

The following is a configuration example of having two spine switches S2 and S3 serving as RPs using phantom RP for redundancy and loadsharing. Here S2 is the primary RP for group-list 227.2.2.0/26 and secondary for group-list 227.2.2.64/26. S3 is the primary RP for group-list 227.2.2.64/26 and secondary RP for group-list 227.2.2.0/26.

Note Phantom RP is used in a PIM BiDir environment where RP redundancy is designed using loopback networks with different mask lengths in the primary and secondary routers. These loopback interfaces are in the same subnet as the RP address, but with different IP addresses from the RP address. (Since the IP address advertised as RP address is not defined on any routers, the term phantom is used). The subnet of the loopback is advertised in the Interior Gateway Protocol (IGP). To maintain RP reachability, it is only necessary to ensure that a route to the RP exists.

Unicast routing longest match algorithms are used to pick the primary over the secondary router.

The primary router announces a longest match route (say, a /30 route for the RP address) and is preferred over the less specific route announced by the secondary router (a /29 route for the same RP address). The primary router advertises the /30 route of the RP, while the secondary router advertises the /29 route. The latter is only chosen when the primary router goes offline. We will be able to switch from the primary to the secondary RP at the speed of convergence of the routing protocol.

Note For ease of use, the configuration mode from which you need to start configuring a task is mentioned at the beginning of each configuration.

Configuration tasks and corresponding show command output are displayed for a part of the topology in the image. For example, if the sample configuration is shown for a leaf switch and connected spine switch, the show command output for the configuration only displays corresponding configuration.

Leaf switch V1 configuration

Phantom Rendezvous-Point association on leaf switch V1

(config) #

```
feature pim
ip pim rp-address 10.254.254.1 group-list 227.2.2.0/26 bidir
ip pim rp-address 10.254.254.65 group-list 227.2.2.64/26 bidir
```

Loopback interface PIM configuration on leaf switch V1

(config) #

```
interface loopback 0
  ip address 10.1.1.54/32
  ip pim sparse-mode
```

IP unnumbered P2P interface configuration on leaf switch V1

(config) #

```
interface Ethernet 1/1
  no switchport
  mtu 9192
  medium p2p
  ip unnumbered loopback 0
  ip pim sparse-mode

interface Ethernet 2/2
  no switchport
  mtu 9192
  medium p2p
  ip unnumbered loopback 0
  ip pim sparse-mode
```

Use an MTU of 9192 for Cisco Nexus 5600 series switches.

Rendezvous Point configuration (on the two spine switches S2 and S3 acting as RPs)

Using phantom RP on spine switch S2

(config) #

```
feature pim
ip pim rp-address 10.254.254.1 group-list 227.2.2.0/26 bidir
ip pim rp-address 10.254.254.65 group-list 227.2.2.64/26 bidir
```

Loopback interface PIM configuration (RP) on spine switch S2/RP1

(config) #

```
interface loopback 0
  ip address 10.1.1.53/32
  ip pim sparse-mode
```

IP unnumbered P2P interface configuration on spine switch S2/RP1 to leaf switch V1

(config) #

```
interface Ethernet 1/1
  no switchport
  mtu 9192
  medium p2p
  ip unnumbered loopback 0
  ip pim sparse-mode
```

Loopback interface PIM configuration (for phantom RP) on spine switch S2/RP1

(config) #

```
interface loopback 1
  ip address 10.254.254.2/30
  ip pim sparse-mode
```

(config) #

```
interface loopback 2
  ip address 10.254.254.66/29
  ip pim sparse-mode
```

Using phantom RP on spine switch S3

(config) #

```
feature pim
ip pim rp-address 10.254.254.1 group-list 227.2.2.0/26 bidir
ip pim rp-address 10.254.254.65 group-list 227.2.2.64/26 bidir
```

Loopback interface PIM configuration (RP) on spine switch S3/RP2

(config) #

```
interface loopback 0
  ip address 10.10.50.100/32
  ip pim sparse-mode
```

IP unnumbered P2P interface configuration on spine switch S3/RP2 to leaf switch V1

(config) #

```
interface Ethernet 2/2
  no switchport
  mtu 9192
  medium p2p
  ip unnumbered loopback 0
  ip pim sparse-mode
```

Loopback interface PIM configuration (for phantom RP) on spine switch S3/RP2

(config) #

```
interface loopback 1
  ip address 10.254.254.66/30
  ip pim sparse-mode


interface loopback 2
  ip address 10.254.254.2/29
  ip pim sparse-mode
```

PIM BiDir Verification

Use the following commands for verifying PIM BiDir configuration:

```
Leaf-Switch-V1# show ip mroute

IP Multicast Routing Table for VRF "default"

(*, 227.2.2.0/26), bidir, uptime: 4d08h, pim ip
  Incoming interface: Ethernet1/1, RPF nbr: 10.1.1.53
  Outgoing interface list: (count: 1)
    Ethernet1/1, uptime: 4d08h, pim, (RPF)

(*, 227.2.2.0/32), bidir, uptime: 4d08h, nve ip pim
  Incoming interface: Ethernet1/1, RPF nbr: 10.1.1.53
  Outgoing interface list: (count: 2)
    Ethernet1/1, uptime: 4d08h, pim, (RPF)
    nve1, uptime: 4d08h, nve

(*, 227.2.2.64/26), bidir, uptime: 4d08h, pim ip
  Incoming interface: Ethernet1/5, RPF nbr: 10.10.50.100/32
  Outgoing interface list: (count: 1)
    Ethernet1/5, uptime: 4d08h, pim, (RPF)

(*, 232.0.0.0/8), uptime: 4d08h, pim ip
  Incoming interface: Null, RPF nbr: 0.0.0.0
  Outgoing interface list: (count: 0)


Leaf-Switch-V1# show ip pim rp

PIM RP Status Information for VRF "default"
BSR disabled
Auto-RP disabled
BSR RP Candidate policy: None
BSR RP policy: None
Auto-RP Announce policy: None
Auto-RP Discovery policy: None

RP: 10.254.254.1, (1),
 uptime: 4d08h   priority: 0,
 RP-source: (local),
 group ranges:
 227.2.2.0/26  (bidir)
RP: 10.254.254.65, (2),
 uptime: 4d08h   priority: 0,
 RP-source: (local),
 group ranges:
 227.2.2.64/26  (bidir)


Leaf-Switch-V1# show ip pim interface

PIM Interface Status for VRF "default"
loopback0, Interface status: protocol-up/link-up/admin-up
  IP address: 10.1.1.54, IP subnet: 10.1.1.54/32
  PIM DR: 10.1.1.54, DR's priority: 1
  PIM neighbor count: 0
  PIM hello interval: 30 secs, next hello sent in: 00:00:23
  PIM neighbor holdtime: 105 secs
  PIM configured DR priority: 1
  PIM configured DR delay: 3 secs
  PIM border interface: no
  PIM GenID sent in Hellos: 0x12650908
  PIM Hello MD5-AH Authentication: disabled
  PIM Neighbor policy: none configured
  PIM Join-Prune inbound policy: none configured
  PIM Join-Prune outbound policy: none configured
  PIM Join-Prune interval: 1 minutes
  PIM Join-Prune next sending: 1 minutes
  PIM BFD enabled: no
  PIM passive interface: no
  PIM VPC SVI: no
  PIM Auto Enabled: no
  PIM Interface Statistics, last reset: never
    General (sent/received):
      Hellos: 13158/0 (early: 0), JPs: 0/0, Asserts: 0/0
      Grafts: 0/0, Graft-Acks: 0/0
      DF-Offers: 0/0, DF-Winners: 0/0, DF-Backoffs: 0/0, DF-Passes: 0/0
    Errors:
      Checksum errors: 0, Invalid packet types/DF subtypes: 0/0
      Authentication failed: 0
      Packet length errors: 0, Bad version packets: 0, Packets from self: 0
      Packets from non-neighbors: 0
          Packets received on passiveinterface: 0
      JPs received on RPF-interface: 0
      (*,G) Joins received with no/wrong RP: 0/0
      (*,G)/(S,G) JPs received for SSM/Bidir groups: 0/0
      JPs filtered by inbound policy: 0
      JPs filtered by outbound policy: 0

Ethernet1/1, Interface status: protocol-up/link-up/admin-up
  IP unnumbered interface (loopback0)
  PIM DR: 10.1.1.54, DR's priority: 1
  PIM neighbor count: 1
  PIM hello interval: 30 secs, next hello sent in: 00:00:04
  PIM neighbor holdtime: 105 secs
  PIM configured DR priority: 1
  PIM configured DR delay: 3 secs
  PIM border interface: no
  PIM GenID sent in Hellos: 0x2534269b
  PIM Hello MD5-AH Authentication: disabled
  PIM Neighbor policy: none configured
  PIM Join-Prune inbound policy: none configured
  PIM Join-Prune outbound policy: none configured
  PIM Join-Prune interval: 1 minutes
  PIM Join-Prune next sending: 1 minutes
  PIM BFD enabled: no
  PIM passive interface: no
  PIM VPC SVI: no
  PIM Auto Enabled: no
  PIM Interface Statistics, last reset: never
    General (sent/received):
      Hellos: 13152/13162 (early: 0), JPs: 2/0, Asserts: 0/0
      Grafts: 0/0, Graft-Acks: 0/0
      DF-Offers: 9/5, DF-Winners: 6249/6254, DF-Backoffs: 0/1, DF-Passes: 0/1
    Errors:
      Checksum errors: 0, Invalid packet types/DF subtypes: 0/0
      Authentication failed: 0
      Packet length errors: 0, Bad version packets: 0, Packets from self: 0
      Packets from non-neighbors: 0
```

```
          Packets received on passiveinterface: 0
      JPs received on RPF-interface: 0
      (*,G) Joins received with no/wrong RP: 0/0
      (*,G)/(S,G) JPs received for SSM/Bidir groups: 0/0
      JPs filtered by inbound policy: 0
      JPs filtered by outbound policy: 0
```

```
Leaf-Switch-V1# show ip pim neighbor

PIM Neighbor Status for VRF "default"

Neighbor     Interface      Uptime    Expires      DR          Bidir-     BFD
                                                  Priority    Capable    State
10.1.1.53    Ethernet1/1    1w1d      00:01:33     1             yes      n/a
10.10.50.100 Ethernet2/2    1w1d      00:01:33     1             yes      n/a
```

For a detailed list of commands, refer to the Configuration and Command Reference guides.

Underlay deployment without multicast (Ingress replication)

Ingress replication is supported on Cisco Nexus 9000 Series switches and not supported on Cisco Nexus 5600 and Cisco Nexus 7000 Series switches.

CHAPTER 6

External Connectivity—MPLS L3VPN

External Connectivity

Introduction to External Connectivity

Before you begin—Ensure you know about Programmable Fabric. Conceptual information is covered in the *Introduction to Cisco Programmable Fabric* and *Introducing Cisco Programmable Fabric (VXLAN/EVPN)* chapters. *Forwarding Configurations* chapter contains Virtual Extensible LAN (VXLAN) BGP EVPN fabric configurations, and the *IP Fabric Underlay* chapter contains unicast and multicast protocol configuration information for the fabric underlay.

VXLAN BGP EVPN fabric supports external connectivity, in that VXLAN BGP EVPN fabric data centers in different sites can be connected using the Data Center Interconnect functionality. Also, VXLAN and non VXLAN pods within a single site can be connected.

You can enable a VXLAN BGP EVPN fabric border leaf switch and an Autonomous System Boundary Router (ASBR) at the edge of the WAN to exchange reachability routes. This way, reachability information is transported between sites. A route exchange scenario at the border leaf or border spine switch is referred as a *handoff* scenario.

Note In this chapter, the focus is on the border leaf switch that acts as a borderPE switch (a border leaf switch with integrated MPLS PE functionality) instead of a border spine switch.

The Cisco Nexus 7000 Series switch is the primary border leaf platform for connecting a VXLAN BGP EVPN fabric to external entities since this switch, with F3 and M3 line cards, acts as the borderPE switch, and provides the combined functionality of a border leaf switch and datacenter edge device. Also, the Cisco Nexus 7000 Series switch provides higher route and VRF scaling capabilities that are required for a border leaf switch.

VM mobility across datacenters

VM mobility across VXLAN BGP EVPN datacenter fabrics works the same way as it does within the datacenter fabric. When VM mobility takes place, the VM generates RARP and GARP messages. You should enable a

Layer-2 DCI such as OTV, Classical Ethernet or VPLS to transport broadcast RARP and GARP packets generated due to the VM movement.

Note Additional configuration is not required to support VM movement across fabrics.

VM mobility across fabrics cannot take place in these scenarios:

1. VM movement takes place when MAC chaining (multiple IP addresses mapped to the same MAC address) is in effect.
2. When an end host sends a non broadcast packet such as ARP on VM move.

Points to consider for connecting two data centers:

- If you want to connect two data center fabrics, restrict the overlay within a data center. Connect two data center instances with an inter data center instance. This way, any instability in one data center will not be spread to the other. Failures can be contained since we are separating the administrative domains.
  - For example, two VXLAN fabrics in different sites are two separate domains, and the MPLS L3VPN inter data center instance forms a different, connecting domain. Here, traffic from the source data center terminates at the border leaf or borderPE switch, and a new Layer-3 inter data center instance sends the traffic to the border leaf switch or borderPE switch of the target datacenter.
- You can create a Layer-2 (OTV, etc) or Layer-3 (LISP, Multiprotocol Label Switching [MPLS] L3VPN, etc) inter datacenter instance.

Layer-3 handoff scenario using MPLS L3VPN is explained in this chapter.

Important Border leaf switch VDCs that have M3 modules do not support LISP handoff scenarios.

Layer 3 hand off scenario – MPLS L3VPN

The VXLAN BGP 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.

Cisco Nexus 7000 Series is considered as the border leaf switch in the MPLS L3VPN scenario explained below. For the MPLS handoff scenario, the border leaf switch is referred to as a borderPE switch. In other words, a border leaf switch with integrated MPLS PE functionality is used.

Note *borderPE switch*—A Cisco Nexus 7000 Series switch with F3/M3 line cards; acts as the collapsed border leaf switch and MPLS Datacenter Provider Edge router. A BorderPE switch includes MPLS PE function. This is also referred to as a one box solution since a single device provides the combined functionality of a border leaf switch and datacenter Edge device.

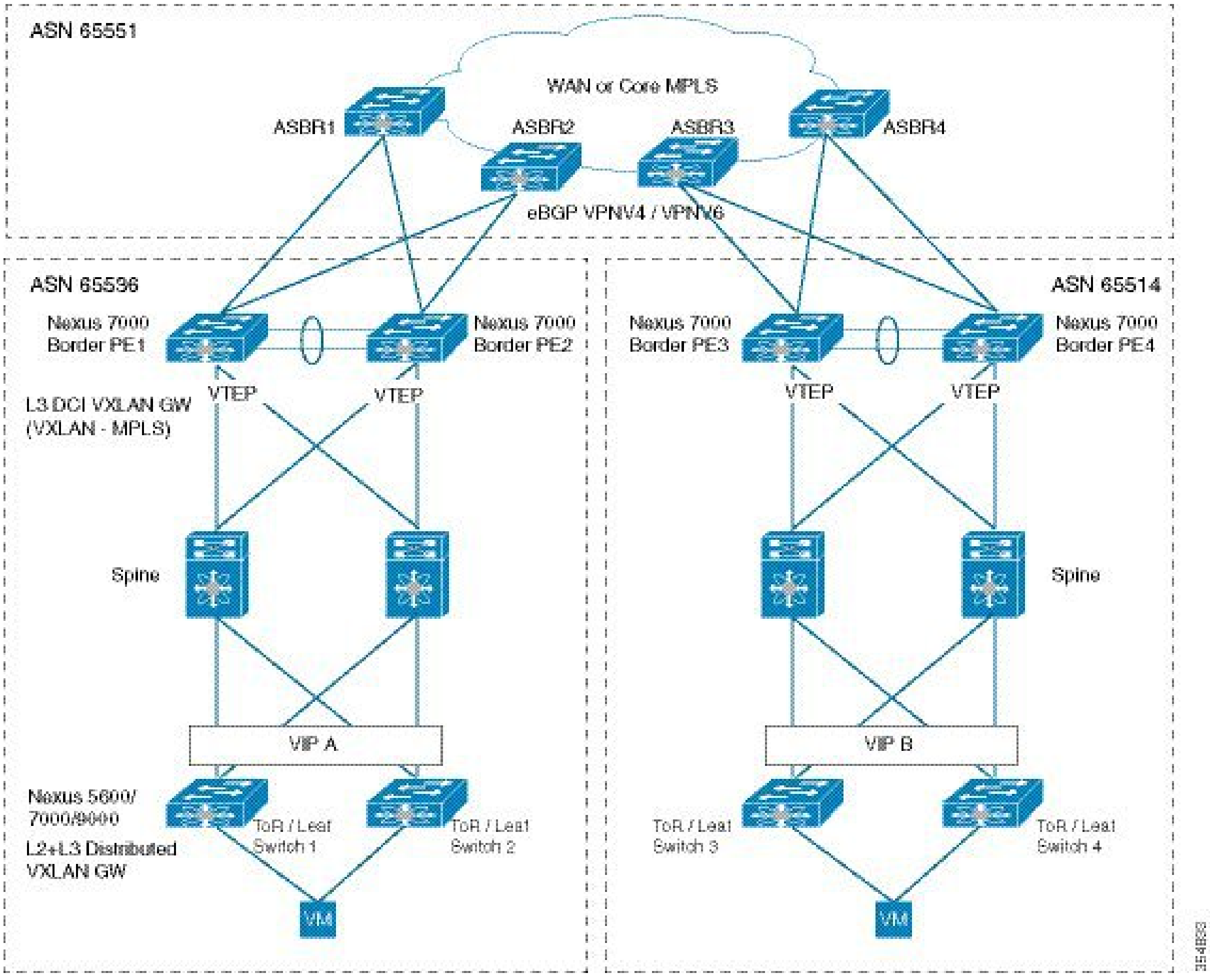
VXLAN BGP EVPN - MPLS L3VPN DCI scenario – In brief

- Two VXLAN BGP EVPN fabric data centers are depicted at the left and the right of the image (below). Routes within a fabric pod are shared between all VTEPs, including with the Cisco Nexus 7000 Series borderPE switches (that are in a vPC setup).
- The borderPE switches and the connected Autonomous System Boundary Routers (ASBRs) of the WAN are configured to pass on routes between each other. For example, routes within the VXLAN BGP EVPN fabric (left) are sent to the WAN ASBR, and (necessary) reachability routes within the WAN are sent to the borderPE switch on the VXLAN fabric (left).
- The WAN ASBR and the VXLAN fabric (right) are also connected in the same way.
- The WAN ASBRs act as MPLS PE nodes that belong to the WAN autonomous system number (ASN), and this ASN is different than that of the 2 VXLAN BGP EVPN fabrics.

As a result, the data center pods depicted in the left and right sides of the image are connected through the WAN using MPLS L3VPN.

Note In addition to connecting data center fabrics, these Layer-3 solutions (MPLS L3VPN, VRF Lite, and LISP) are also used to connect campus networks.

Figure 37: DCI using MPLS L3VPN



VXLAN BGP EVPN - MPLS L3VPN DCI scenario - In more detail

Route distribution and data flow between the VXLAN pod and the WAN MPLS is explained below.

Route distribution within the VXLAN pod, and subsequent export of VXLAN pod routes to the WAN MPLS

The routes within the VXLAN BGP EVPN pod should be exported to the WAN MPLS, thereby extending Layer-3 reachability from the WAN MPLS to the ToRs within the VXLAN BGP EVPN fabric.

- The BGP EVPN control plane in the VXLAN BGP EVPN fabric ensures distribution of routes between ToR/leaf switch VTEPs (including the borderPE switch) within the fabric. ToRs will forward the attached end host IP and MAC addresses, /32 (or /128, for IPv6 addresses) ‘Host IP + MAC’ routes, and Layer-2 and Layer-3 VXLAN VNIs using the EVPN Route Type 2/5 option. Based on the import route target (RT) configured on the border leaf switch, the switch will import the /32 (or /128) routes into appropriate VRF tables.
- The borderPE switch advertises the VRF default route to the ToR/leaf switches. If the ToR/leaf switch nodes receive the same route from multiple borderPE switches, it results in ECMP at the ingress ToR/leaf switch.

- The borderPE switch should be configured to *reoriginate* the /32 (or /128) routes on an eBGP VPNv4 session towards the WAN MPLS. The borderPE switch does MPLS switching towards the WAN MPLS and VXLAN BGP EVPN routes are reoriginated into the L3VPN address family.

VXLAN fabric to WAN Data flow—Let us say a host in the VXLAN fabric (left) sends traffic to a host in the other VXLAN fabric (right). A high level data plane flow is depicted below:

- The VXLAN packet reaches the borderPE switch.
- A VXLAN VNI lookup happens. This lookup maps to the appropriate bridge domain.
- The subsequent MAC address lookup points to a bridge domain interface on the borderPE switch, and the packet is bridged to the BDI interface. The BDI interface then points to the corresponding VRF table.

Note The BDI is used for VRF routing and has no IP.

- The destination IP address is checked in the VRF IP table, wherein the corresponding L3VPN MPLS encapsulation towards the WAN MPLS is pointed at.
- The packet is sent to the WAN ASBR.

Importing of WAN or external routes into the VXLAN BGP EVPN borderPE switch

After WAN or external routes are imported into the borderPE switch, they are re-advertised to the ToR/leaf switches in the VXLAN BGP EVPN fabric with the borderPE VTEP acting as the next hop for the ToR/leaf switches, thereby extending Layer-3 reachability from the ToR/leaf switches to the WAN.

- The borderPE (Cisco Nexus 7000 Series) switch will function as an EVPN based Layer-3 VXLAN Gateway to provide an overlay routing function for Layer-3 IP traffic between the VXLAN overlay fabric and the WAN. This includes Layer-3 flows between the VPNv4/6 and/or IP end points outside the VXLAN pod (like hosts in the other data center, etc).
- The borderPE switch receives routes from L3VPN ASBRs to enable connectivity to the WAN. This is achieved through an external BGP VPNv4 or VPNv6 session with the WAN ASBR device. The L3VPN routes from the WAN will be imported into local VRF tables in the borderPE switch.
- The borderPE switch should be configured to re-originate the L3VPN routes into the EVPN address family so as to send the routes via BGP EVPN control plane from the borderPE switch towards the ToR/leaf switches. The ToR/leaf switches will import these routes into the appropriate VRF.
- Necessary configuration knobs will need to be added in BGP under **VRF**, and under **neighbor evpn** address family to originate a default route towards EVPN neighbors and drop all other routes.

WAN to VXLAN BGP EVPN fabric Data flow—Traffic from the WAN ASBR arrives at the borderPE switch. A high level flow is depicted below:

- The packet arrives with a local, per VRF VPN label (advertised earlier by the borderPE switch). The MPLS label lookup points to the correct VRF table.
- The (VRF, IP) lookup results in a /32 (or /128) route that points to the VXLAN tunnel end point adjacency for the ToR/leaf switch VTEP on the fabric facing per VRF BDI interface.
- The packet is VXLAN encapsulated with the router MAC (RMAC) address of the remote ToR/leaf switch VTEP and sent on the VRF BDI interface.

- The FIB lookup drives the VXLAN encapsulation towards the designated remote ToR/leaf switch VTEP.

The fabric is stitched to the VPN service

- Once the tenant flows within the 2 data centers are stitched to the IP VPN service, routes from data center (left) are connected to data center (right) and vice versa.

BorderPE switches in a vPC setup

The two borderPE switches are configured as a vPC. In a VXLAN vPC deployment, a common, virtual VTEP IP address (secondary loopback IP address) is used for communication. The common, virtual VTEP uses a system specific router MAC address. The Layer-3 prefixes or default route from the borderPE switch will be advertised with this common virtual VTEP as the next hop, and the ToR/leaf switch VTEPs will install the default route and/or prefix route with a single BGP path to the borderPE common, virtual VTEP IP address.

Advertising primary IP address (PIP)

On a vPC enabled leaf or border leaf switch, by default all Layer-3 routes are advertised with the secondary IP address (VIP) of the leaf switch VTEP as the BGP next-hop IP address. Prefix routes and leaf switch generated routes are not synced between vPC leaf switches. Using the VIP as the BGP next-hop for these types of routes can cause traffic to be forwarded to the wrong vPC leaf or border leaf switch and black-holed. The provision to use the primary IP address (PIP) as the next-hop when advertising prefix routes or loopback interface routes in BGP on vPC enabled leaf or border leaf switches allows users to select the PIP as BGP next-hop when advertising these types of routes, so that traffic will always be forwarded to the right vPC enabled leaf or border leaf switch.

The configuration command for advertising the PIP is **advertise-pip**.

A sample configuration is given below:

(config) #

```
router bgp 65536
  address-family l2vpn evpn
      advertise-pip
      advertise-system-mac
```

The **advertise-pip** command lets BGP use the PIP as next-hop when advertising prefix routes or leaf generated routes if vPC is enabled. The **advertise-system-mac** command lets BGP advertise Route-type-2 routes that includes VIP and router-mac information. This is needed for solving an issue in EVPN decapsulation on remote leaf switches when PIP is used as next-hop in the BGP advertisement.

BorderPE switch to WAN ASBR link failure scenario

If there is link failure between one of the two borderPE switches and the connected L3VPN ASBR, the switch will withdraw the BGP routes that are being advertised towards the fabric and traffic re-convergence happens through the redundant border leaf switch.

For border leaf switches (in a two box solution), the default route will be withdrawn when both links to the DC Edge router fail. For a borderPE switch, advertising default route is not recommended.

Configuration for the VXLAN BGP EVPN - MPLS L3VPN DCI scenario

The following configurations enabled on the borderPE switch establish a Layer-3 link along with the MPLS/LDP configuration between the borderPE switch and the WAN ASBR. After configurations on the borderPE switch and the WAN ASBR device, routes are exchanged between the VXLAN fabric borderPE switch and the MPLS WAN ASBR.

Important Ensure that you follow these implementation pointers:

- This document only contains Cisco Nexus 7000 Series borderPE switch related configurations. To complete Layer-3 DCI configurations, you should also enable corresponding configurations on the WAN ASBR.
- To forward traffic across the borderPE switch (from the VXLAN BGP EVPN fabric towards the WAN ASBR and the other way round), the **fabric forwarding switch-role border** command should be mandatorily configured on the borderPE switch. Since the change of switch role requires a switch reload (through **write erase** and **reload** commands), ensure that this command is included in the startup configuration. For the borderPE Layer-3 extension auto configuration feature, use the **fabric forwarding switch-role border dci-node** command.
- The physical interface connecting the VXLAN BGP EVPN fabric should be different from the IP/MPLS WAN facing interface. The same physical Layer-3 interface or sub interface should not be used to connect the VXLAN BGP EVPN Fabric and the WAN Core.
- For an F3 only VDC, the Layer-3 backup link that is used to protect the WAN facing interfaces from failure should not be the peer link or SVI of the VLANs extended over the peer link. *A separate Layer-2 interface with a dedicated VLAN/SVI, or a separate Layer-3 interface or sub interface should be used.*
- For an M3-F3 VDC, traffic received from the VXLAN BGP EVPN fabric should not be forwarded to a Layer-3 sub interface. VXLAN terminated traffic can only be forwarded over a Layer-3 physical interface.
- Host Mobility Manager (HMM) CLIs are removed with the **no feature fab forwarding** command even though the **nv overlay evpn** command is present.

  The **feature fabric forwarding** command is not needed if **nv overlay evpn** is already configured.

On BorderPE1 switch, enable VXLAN BGP EVPN features

(config) #

```
install feature-set fabric
feature-set fabric
feature fabric forwarding
feature interface-vlan
feature ospf  (OR feature isis)
feature nv overlay
feature bgp
feature vni
nv overlay evpn
install feature-set mpls
feature-set mpls
feature mpls l3vpn
feature mpls ldp
```

The VXLAN feature related configurations shown above are already enabled on all switches in the VXLAN BGP EVPN fabric. This has been included here included for completeness only.

Note The **install feature-set fabric** command should only be used in the admin VDC. When using a VDC, ensure the VDC is of type F3 or M3, for EVPN. A sample configuration is given below:

(config) #

```
vdc BorderPE1
   limit-resource module-type f3
```

Configure the anycast gateway MAC address

(config) #

```
fabric forwarding anycast-gateway-mac 0202.0002.0002
```

On BorderPE1, configure a bridge domain and associate a Layer-3 network VNI

(config) #

```
fabric forwarding switch-role border
system bridge-domain 2500-3500
system fabric bridge-domain 2500-2999
vni 31000
bridge-domain 2500
   member vni 31000
```

On BorderPE1, create a VRF and associate the previously configured VNI to it. Then, enable importing and exporting of routes between the VXLAN BGP EVPN fabric and the MPLS WAN side, and create the default routes to be injected into the VXLAN fabric

(config) #

```
vrf context vni-31000
  vni 31000
  rd auto
  address-family ipv4 unicast
      route-target import 65551:1
      route-target export 65551:1
      route-target both auto
      route-target both auto evpn
```

Type **exit** and configure IPv6 route import/export.

```
  address-family ipv6 unicast
     route-target import 65551:1
     route-target export 65551:1
     route-target both auto
     route-target both auto evpn
```

65551:1 refers to the import/export of WAN routes.

Within the VXLAN BGP EVPN fabric, NX-OS automatically assigns the correct route target. It is recommended that the commands **route-target both auto** and **route-target both auto evpn** are used on the ToR/Leaf switches too.

Note The route target *65551:1* should be the same as the route target configured on the connected WAN ASBR since the importing/exporting of MPLS information is based on this route target

By using route targets as the glue, the BGP EVPN control plane (in the VXLAN fabric) and the BGP L3VPN control plane (from the fabric to the WAN) are connected. Similarly, for a tenant VRF, the same route target should be enabled on ToR/leaf switches and the border leaf switch(es).

On BorderPE1, configure a bridge domain and BDI for the VRF

(config) #

```
interface BDI 2500
   no shutdown
   mtu 9192
   vrf member vni-31000
   no ip redirects
   ip forward
   ipv6 address use-link-local-only
   ipv6 forward
   no ipv6 redirects
```

After the above configuration, we ensure that a bridge domain interface is designated for Layer-3 traffic transportation.

Important The **interface BDI** configuration is not required to be configured manually, when the profile vrf-tenant-profile is configured. As soon the VRF context is configured, NX-OS automatically calls the profile vrf-tenant-profile and applies appropriate configurations. vrf-tenant-profile is always added when you deploy the BorderPE switch with POAP from DCNM. The resulting interface BDI configuration can be verified with the **show run inter bdi 2500 expand-port-profile** command.

The vrf-tenant-profile configuration is given below:

```
configure profile vrf-tenant-profile
vni $vrfSegmentId
bridge-domain $bridgeDomainId
member vni $vrfSegmentId
interface bdi $bridgeDomainId
vrf member $vrfName
ip forward
no ip redirects
ipv6 forward
ipv6 address use-link-local-only
no ipv6 redirects
mtu 9192
no shutdown
```

On BorderPE1, add the Layer-3 VRF VNI to the overlay

(config) #

```
interface nve 1
   no shutdown
   source-interface loopback 1
   host-reachability protocol bgp
   member vni 31000 associate-vrf
```

In the above configuration, we ensure that the Layer-3 VNI is associated with the VXLAN VTEP.

On BorderPE1, establish a multihop external BGP session to the WAN ASBR and enable forwarding of L2VPN routes towards the WAN ASBR

(config) #

```
router bgp 65536
   neighbor 209.165.200.225 remote-as 65551
      update-source loopback 100 -> Optional
      ebgp-multihop 10 -> Optional
      address-family vpnv4 unicast
        send-community both
        import l2vpn evpn reoriginate
```

If the BGP session to the WAN ASBR is on a directly connected interface, and the peering is done on the interface address, then the **ebgp-multihop** and **update-source** commands are not required. Also, configure the VPNv6 address family as shown below:

```
     address-family vpnv6 unicast
        send-community both
        import l2vpn evpn reoriginate
```

Attention In the above configurations, the L2VPN EVPN information is being imported into the VPNv4/VPNv6 address families so that the routes in the VXLAN BGP EVPN fabric can be sent over VPNv4/VPNv6 to the connected WAN ASBR. When the WAN ASBR sends routes to the border leaf switch, the received VPNv4/VPNv6 routes need to be sent into the VXLAN BGP EVPN control plane. To achieve that, the VPNv4/VPNv6 information (L3VPN routes) is imported into the L2VPN EVPN address family, as shown below.

Configure the BGP EVPN neighbor within the fabric.

(config) #

```
router bgp 65536
   neighbor 10.2.2.1 remote-as 65536
      update-source loopback 0
      address-family l2vpn evpn
         send-community both
         import vpn unicast reoriginate
```

On BorderPE1, create the VRF under the BGP configuration to advertise the L2VPN EVPN address family (routes) within the VRF

(config) #

```
router bgp 65536
  vrf vni-31000
     address-family ipv4 unicast
       advertise l2vpn evpn
       maximum-paths 2
       label-allocation-mode per-vrf
       exit
     address-family ipv6 unicast
       advertise l2vpn evpn
       maximum-paths 2
       label-allocation-mode per-vrf
```

Note Alternatively, you can enable a 0/0 default route origination in each tenant VRF (on the border leaf switch). The ToRs/leaf switches will import the default route, resulting in a VRF default route towards the border leaf switch in the ToR switch tenant VRFs.

Creation of default route and route maps on the borderPE1 switch

Enable default route origination in each VRF, for IP4 and IPv6 address families

(config) #

```
vrf context vni-31000
  ip route 0.0.0.0/0 null 0 254
  ipv6 route 0::/0 null 0 254
```

The ToR switches will import this default route, resulting in a VRF default route for tenant VRF *vni-31000* on the ToR switches.

Create a route-map to ensure that the default route from the WAN ASBR is preferred over a default route from other sources

(config) #

```
route-map PREFER-EXTERNAL-DEFAULT permit 100
   set local-preference 50
```

Apply the route map to tenant VRF vni-31000 address families (IP4 and IPv6 unicast)

(config) #

```
vrf vni-31000
     address-family ipv4 unicast
       network 0.0.0.0/0 evpn route-map PREFER-EXTERNAL-DEFAULT
       exit
     address-family ipv6 unicast
       network 0.0.0.0/0 evpn route-map PREFER-EXTERNAL-DEFAULT
```

Restrict default routes generated or learned on the BorderPE switch from being distributed to the external WAN ASBR

(config) #

```
 ip prefix-list default-route seq 5 permit 0.0.0.0/0 le 1
 ipv6 prefix-list default-route-v6 seq 5 permit 0::0/0
 route-map DENY-DEFAULT-ROUTE deny 10
    match ip address prefix-list default-route
    exit
 route-map DENY-DEFAULT-ROUTE permit 1000
 exit
 route-map DENY-DEFAULT-ROUTE-v6 deny 100
   match ipv6 address prefix-list default-route-v6
   exit
 route-map DENY-DEFAULT-ROUTE-v6 permit 1000
```

BGP specific default route configurations

Restrict default routes generated in the fabric from being distributed to external neighbors through BGP

(config) #

```
router bgp 65536
   neighbor 209.165.200.225 remote-as 65551
      address-family vpnv4 unicast
         route-map DENY-DEFAULT-ROUTE out
         exit
      address-family vpnv6 unicast
         route-map DENY-DEFAULT-ROUTE-v6 out
```

After the above configuration, we ensure that default routes will not be included in the VPNv4 and VPNv6 routes sent to the WAN ASBR.

Glossary

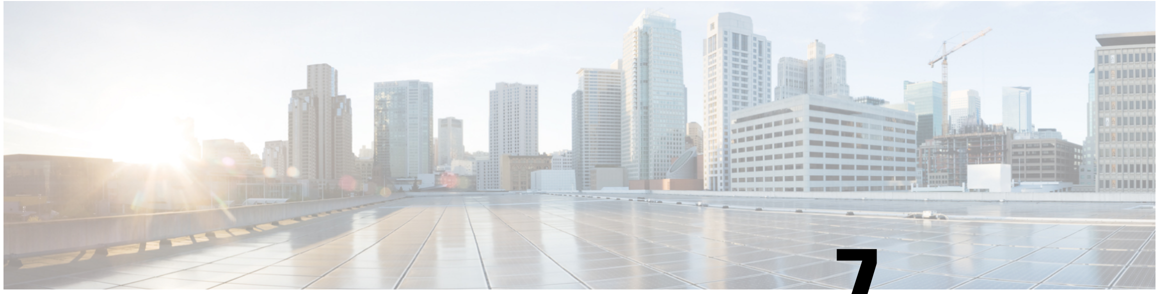
RD—Route Distinguisher

RT—Route Target

RR—BGP route reflector. A route reflector reflects incoming routes to all other leaf switch nodes. Typically, spine switches are configured as route reflectors.

BorderPE switch—A Cisco Nexus 7000 Series switch with an F3 or M3 line card; acts as the collapsed border leaf switch and Cisco Nexus 7000 Series data center edge switch. A BorderPE switch includes the MPLS PE function. This is also referred to as a one box solution.

A two box solution comprises of two switches (A Cisco Nexus 5600 Series or 7000 Series border leaf switch + a Cisco Nexus 7000 Series data center edge switch) to route IP frames from an external network into the VXLAN BGP EVPN fabric



CHAPTER 7

External Connectivity—VRF Lite

External Connectivity

Layer-3 DCI for VXLAN BGP EVPN fabrics—VRF lite

VXLAN BGP EVPN data center fabrics can be connected across Layer-3 boundaries using MPLS L3VPN, VRF IP Routing (VRF lite), or LISP as the mechanism of transport outside the VXLAN fabric. The VRF lite scenario is explained in this chapter.

VM mobility across datacenters

VM mobility across VXLAN BGP EVPN datacenter fabrics works the same way as it does within the datacenter fabric. When VM mobility takes place, the VM generates RARP and GARP messages. You should enable a Layer-2 DCI such as OTV, Classical Ethernet or VPLS to transport broadcast RARP and GARP packets generated due to the VM movement.

Note Additional configuration is not required to support VM movement across fabrics.

VM mobility across fabrics cannot take place in these scenarios:

1. VM movement takes place when MAC chaining (multiple IP addresses mapped to the same MAC address) is in effect.

2. When an end host sends a non broadcast packet such as ARP on VM move.

VXLAN BGP EVPN - VRF lite scenario - In brief

Some pointers are given below:

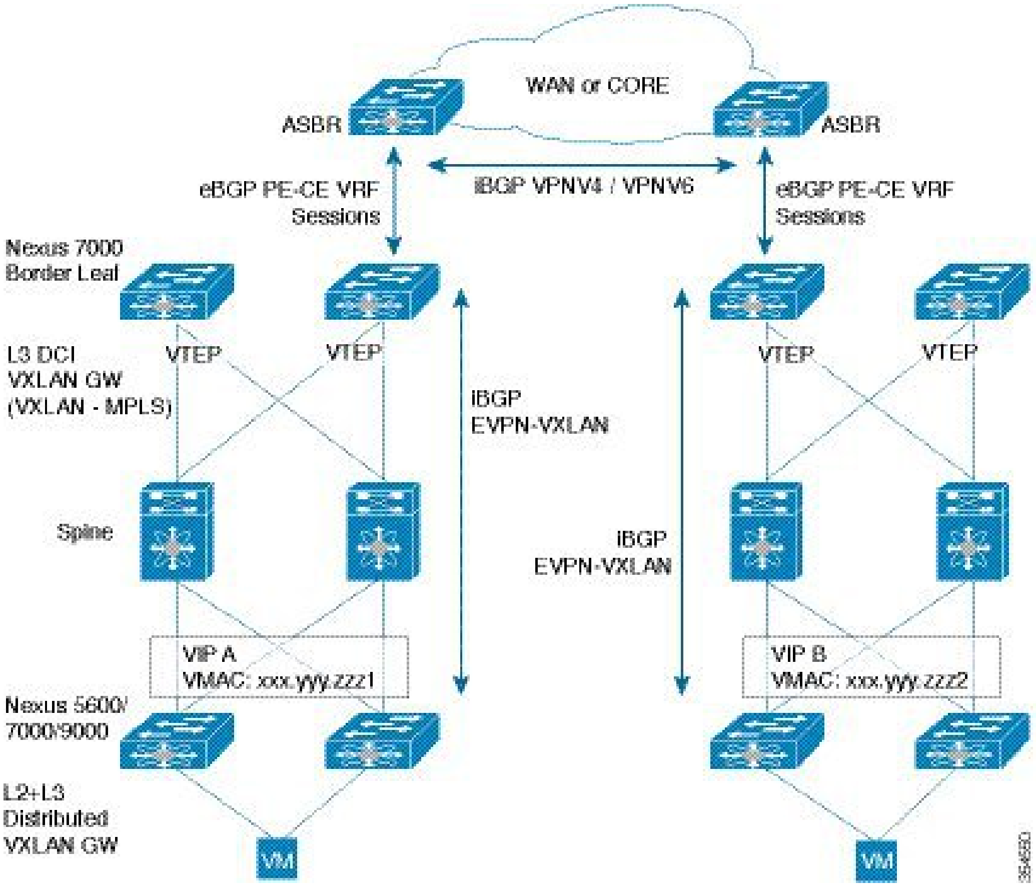
- The VXLAN BGP EVPN fabric pods are depicted at the left and the right of the below image (*VRF Lite DCI hand-off topology*). Routes within a fabric pod are shared between all VTEPs within the pod, including with the Cisco Nexus 7000 Series border leaf switches.
- The border leaf switches and the WAN ASBR/PE routers are configured to pass on routes between each other through eBGP sessions, using VRF Lite.

For example, tenant VRF routes within VXLAN BGP EVPN fabric (left) are sent to the WAN ASBR routers, and (necessary) reachability routes within the WAN are sent to the border leaf switches on the VXLAN fabric (left).

- The WAN and the VXLAN fabric (right) can also be connected in a similar way.

As a result, the data center pods depicted in the left and right of the image are seamlessly connected through the WAN using VRF lite.

Figure 38: VRF Lite DCI hand-off topology



VXLAN BGP EVPN – VRF lite scenario – In more detail

Route distribution between the VXLAN pod (left) and the WAN are explained in the order given below:

1. Route distribution within the VXLAN pod, and subsequent export of VXLAN pod routes from the Border Leaf switch to the WAN.
2. Importing of VXLAN BGP EVPN fabric routes on the WAN edge device.
3. Importing of WAN routes into the VXLAN BGP EVPN border leaf switch.

Step 1 - Route distribution within the VXLAN pod, and subsequent export of VXLAN pod routes to the WAN

The routes within the VXLAN BGP EVPN pod can be exported to the WAN by the following process, thereby extending Layer-3 reachability from the WAN to the ToRs in the VXLAN BGP EVPN fabric.

- The BGP EVPN control plane in the VXLAN BGP EVPN fabric ensures distribution of routes between VTEPs (and to the border leaf switches) within the fabric. ToRs will forward the attached host IP and MAC addresses (/32 Host IP [or /128, for IPv6 addresses] + MAC routes) using the EVPN Route Type 5 option and the border leaf switch will import the /32 (or /128) routes into local VRF tables.

Note For VRF Lite extension, configure only those VRF instances on the border leaf switch that need to be extended outside the fabric. You need to configure a Layer-3 sub interface towards the WAN ASBR/PE and establish an eBGP session over that. Based on the configurations on the border leaf switch, the data is forwarded towards the WAN.

- If configured to do so, the border leaf switch advertises a 0/0 default route (IPv4/IPv6) per VRF to the other leaf switches/ToRs. When ToR switch nodes receive the same route from multiple border leaf switches, it results in ECMP at the ingress ToR/leaf switch nodes.

Step 2 - Importing of VXLAN BGP EVPN fabric routes on the WAN ASBR/PE device

- To receive VXLAN fabric routes from the border leaf switch, the WAN ASBR/PE routers should also have Layer-3 sub interfaces configured. The 802.1Q ID on each sub interface needs to be the same for a tenant on the WAN ASBR/PE router and the fabric border leaf switch.
- Similar configurations need to be implemented on the WAN ASBR/PE router (right) designated to communicate with the border leaf switch(es) on the VXLAN fabric (right).
- The configurations on the WAN ASBR/PE routers and the respective border leaf switches ensure that fabric routes are sent to the WAN and WAN routes to the fabric, for the configured VRFs.

Step 3 -Importing of WAN ASBR/PE routes into the VXLAN BGP EVPN border leaf switch

Routes arriving at the border leaf switch need to be re-advertised to the ToRs in the VXLAN BGP EVPN fabric, with the border leaf VTEP as the next hop for the ToR switches. As a result, Layer-3 reachability is extended from the ToR switches to the WAN. The process is given below.

- The WAN routes arrive at the relevant sub interfaces on the border leaf switch.
- The border leaf switches can be configured to re-originate these imported VRF prefixes towards the EVPN control plane (on the fabric side) or can be configured to originate a 0/0 default route in each VRF. If a default route is configured, the ToRs/leaf switches will import this default route, resulting in a VRF default route towards the border leaf switches in all relevant tenant VRFs on the ToRs.
- If configured to distribute default routes, necessary configuration knobs need to be added in BGP under **VRF**, and under the **neighbor evpn** address family to originate a default route towards EVPN neighbors and drop all other routes.

Data Flow

Data flow for Layer-3 DCI using VRF Lite is given below:

Route distribution within the VXLAN pod, and export of VXLAN pod routes to the WAN

Let us say a host in the VXLAN fabric (left) initiates communication with a host in the other VXLAN pod (right). A high level data plane flow is depicted below:

- The VXLAN packet reaches the border leaf switch. If a default route is configured, (the leaf switch uses the default IP route pointing to the BL switch as the next hop).

Note For the leaf switch, any destination outside the fabric will have the VTEP of the border leaf as the next hop.

- A VXLAN VNI lookup happens. This lookup points to the appropriate bridge domain and VRF interface.
- The IP lookup in the VRF IP table points to the WAN IP adjacency on the VRF interface.
- The IP packet is routed to the WAN PE or ASBR.

Importing of WAN ASBR/PE routes into the VXLAN BGP EVPN border leaf switch

Traffic from the WAN ASBR/PE device arrives at the border leaf switch on corresponding sub interfaces. A high level flow is depicted below:

- The IP packet arrives with the 802.1Q tag associated with the sub interface. The sub interface points to the correct VRF table.
- The (VRF, IP) lookup results in a /32 (or /128) route that points to the VXLAN tunnel end point adjacency for the ToR VTEP, on the fabric facing per VRF BDI interface.
- The packet is (VXLAN) encapsulated with the router MAC (RMAC) address of the remote ToR VTEP and sent on the VRF BDI interface, and the FIB lookup drives the VXLAN encapsulation towards the designated remote ToR VTEP.

The VXLAN fabrics are stitched together

- Once the VXLAN BGP EVPN fabric routes of the left and right datacenters are exchanged between each other, connectivity is complete.

Border leaf switches in a vPC scenario

In a vPC scenario, the two border leaf switches are configured for a Layer-2 handoff scenario, and a common, virtual VTEP IP address is used for communication. To use the same VTEP for Layer-2 and Layer-3 handoff scenarios, a common VTEP IP address and router MAC address should be configured on the border leaf switches. The Layer-3 prefixes or default route from the border leaf switch will be advertised with this virtual VTEP as the next-hop, and the ToR VTEPs will install the default route or prefix route with a single BGP path to the border leaf (virtual) VTEP.

Border leaf to WAN ASBR/PE link failure scenario

If there is a link failure between the designated border leaf switch and the WAN ASBR/PE device, the switch will withdraw the BGP routes that are being advertised towards the fabric and traffic re-convergence happens through the redundant border leaf switch.

Attention Remove the default route manual configuration on the failed switch interface. Else, BGP re-convergence will fail since a default route will originate from both the border leaf switches.

Configuration for the VXLAN BGP EVPN—VRF lite scenario

Typically, most of the VXLAN BGP EVPN configuration is done during initial configuration of the fabric. Refer to the chapters *Forwarding concepts* and *IP Fabric Underlay* for more details.

On border leaf switch 1, configure a bridge domain and associate a Layer-3 network VNI

(config)#

```
system bridge-domain 2500-3500
system fabric bridge-domain 2500-2999
vni 50000
bridge-domain 2500
   member vni 50000
```

Create a VRF and associate the Layer-3 VNI to the VRF. Then, enable importing and exporting of VXLAN fabric routes on the border leaf switch

(config)#

```
vrf context vni-50000:p1
  vni 50000
  rd auto
   address-family ipv4 unicast
     route-target both auto
     route-target both auto evpn
   address-family ipv6 unicast
     route-target both auto
     route-target both auto evpn
```

Configuration towards the WAN

Create a subinterface and associate an 802.1Q tag to it

(config)#

```
interface port-channel 1.100
```

Note You can implement configurations on interfaces, that is, associate an interface to an 802.1Q tag.

(config-subif)#

```
encapsulation dot1q 100
vrf member vni-50000:p1
ip address 192.0.2.1/24
ipv6 address 2001:DB8::1
```

The 802.1Q tag (100) acts as the distinguisher that tells the border leaf switch which VRF the packet belongs to.

Configure BGP and associate the tenant VRF under BGP

(config)#

```
router bgp 65536
  vrf vni-50000:p1
    address-family ipv4 unicast
      advertise l2vpn evpn
      maximum-paths ibgp 2
      maximum-paths 2
    address-family ipv6 unicast
      advertise l2vpn evpn
      maximum-paths ibgp 2
      maximum-paths 2
    neighbor 192.0.2.3 remote-as 65551
       address-family ipv4 unicast
        send-community both
       address-family ipv6 unicast
        send-community both
```

65551 is the BGP autonomous system (AS) ID of the WAN.

Add the Layer-3 VRF VNI to the overlay

(config)#

```
interface nve 1
  source-interface loopback 0
  member vni 50000 associate-vrf
    host-reachability protocol bgp
```

Configure a BDI and associate the Layer-3 VRF to it

(config)#

```
interface BDI 2500
   ip forwarding enable
   vrf member vni-50000:p1
```

Establish a BGP session within the VXLAN BGP EVPN fabric to enable L2VPN routes' distribution

(config)#

```
router bgp 65536
   neighbor 10.2.2.1 remote-as 65536
     address-family l2vpn evpn
```

Attention The above configurations contain VRF Lite configuration without default route creation. To create default routes and distribute them to the leaf/ToR switches, add the following configurations

Enable default route distribution

Configure IPv4 and IPv6 default routes for the corresponding VRF

(config)#

```
vrf context vni-50000:p1
  vni 50000
  rd auto
    ip route 0.0.0.0/0 192.0.2.3
    ipv6 route 0::/0 2001:DB8:1::1
```

Note The **ip route** and **ipv6 route** commands are specific to default route configuration.

Create a prefix list to enable IP prefix filtering

(config)#

```
ip prefix-list default-route seq 5 permit 0.0.0.0/0 le 1
ipv6 prefix-list default-route-v6 seq 5 permit 0::/0
route-map DENY-DEFAULT-ROUTE deny 10
  match ip address prefix-list default-route
```

Create route maps to restrict default routes generated in the fabric from being distributed to external neighbors

(config)#

```
route-map DENY-DEFAULT-ROUTE permit 1000
route-map DENY-DEFAULT-ROUTE-V6 deny 100
  match ipv6 address prefix-list default-route-v6
route-map DENY-DEFAULT-ROUTE-V6 permit 1000
route-map FABRIC-RMAP-REDIST-STATIC permit 10
  match ip address prefix-list default-route
route-map FABRIC-RMAP-REDIST-STATIC-V6 permit 100
  match ipv6 address prefix-list default-route-v6
```

The **DENY-DEFAULT-ROUTE** route map is created to restrict default routes from being sent to external networks

Enable BGP configuration and apply appropriate route maps

(config)#

```
router bgp 65536
  vrf vni-50000:p1
    address-family ipv4 unicast
      redistribute static route-map FABRIC-RMAP-REDIST-STATIC
      default-information originate
    address-family ipv6 unicast
      redistribute static route-map FABRIC-RMAP-REDIST-STATIC-V6
      default-information originate
  neighbor 192.0.2.3 remote-as 65551
    address-family ipv4 unicast
      send-community both
      route-map DENY-DEFAULT-ROUTE out
    address-family ipv6 unicast
      send-community both
      route-map DENY-DEFAULT-ROUTE-V6 out
```

Note: The **redistribute static route-map**, **default-information originate**, and **route-map** commands are specific to default route configuration.

Some pointers for the above configurations are given below:

- The configured next hop IPv4 static route 192.0.2.3 (and next hop IPv6 route 2001:DB8:1::1) of the 0/0 default route is the IP address of the WAN ASBR/PE local interface facing the border leaf switch. By pointing the static route towards the interface facing the WAN routers, it is ensured that the default route will be withdrawn if the interface(s) facing the WAN routers are down.
- The default IP route configuration needs to be enabled only on the border leaf switch nodes, and not on any other leaf switch.
- The **redistribute static route-map** and **default-information originate** commands enable distribution of IPv4 and IPv6 default routes to ToR/leaf switches.
- *65551* is the BGP autonomous system (AS) ID of the WAN.

CHAPTER 8

External Connectivity—LISP

External Connectivity

Layer-3 DCI for VXLAN BGP EVPN fabrics—LISP

VXLAN BGP EVPN data center fabrics can be connected across Layer-3 boundaries using MPLS L3VPN, VRF IP Routing (VRF lite), or LISP as the mechanism of transport outside the VXLAN fabric. The LISP scenario is explained in this chapter.

Some pointers about LISP as the data center interconnect (DCI) technology is given below:

- A Cisco Nexus 7000 (or 7700) Series switch (with F3/M3 line cards) is considered as the border leaf switch in the LISP scenario explained below.
- F3/M3 line cards provide multiple data plane encapsulation in hardware and control plane protocols. VXLAN encapsulation is implemented in hardware on the southbound side, and LISP is implemented in hardware on the northbound side of the F3/M3 cards.

  Note For the F3 line card, support was introduced in the Cisco NX-OS Software Release 7.2. For the M3 line card, support was introduced in the 8.2(1) release.

- LISP provides an optimal way to resolve ingress route optimization challenges that result from workload mobility across data centers.
- Unlike other traditional routing protocols, LISP is a PULL based model and only interested entities (switches or routers) ask for information. The PULL based model results in a smaller state in the switch tables and conserves hardware space.
- LISP has other advantages, as noted below:
  - Mobility—IP prefix address family portability.
  - Scalability—On demand routing.
  - Security—Tenant ID based segmentation.

VM mobility across datacenters

VM mobility across VXLAN BGP EVPN datacenter fabrics works the same way as it does within the datacenter fabric. When VM mobility takes place, the VM generates RARP and GARP messages. You should enable a Layer-2 DCI such as OTV, Classical Ethernet or VPLS to transport broadcast RARP and GARP packets generated due to the VM movement.

Note Additional configuration is not required to support VM movement across fabrics.

VM mobility across fabrics cannot take place in these scenarios:

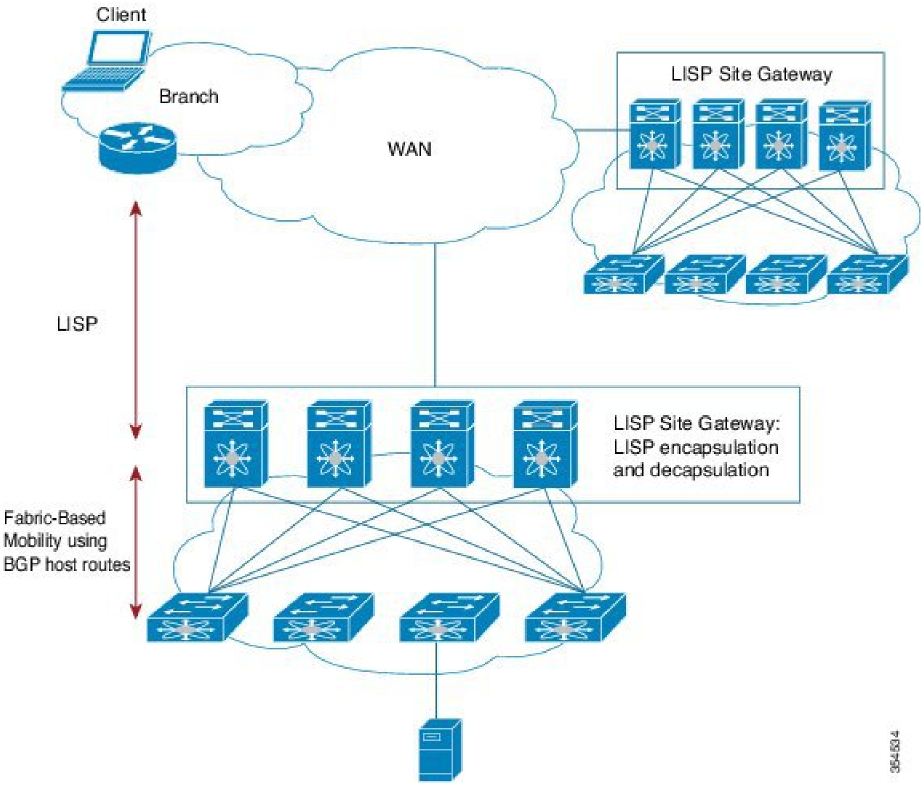
1. VM movement takes place when MAC chaining (multiple IP addresses mapped to the same MAC address) is in effect.
2. When an end host sends a non broadcast packet such as ARP on VM move.

Host Mobility across VXLAN BGP EVPN Fabrics using LISP

Host move detection within the fabric

In the VXLAN BGP EVPN fabric, the host routes and MAC address information are distributed in the MP-BGP EVPN control plane, which means that the fabric itself performs the host detection. The LISP site gateways use these host routes for triggering the LISP mobility encapsulation and decapsulation. LISP, when integrated with the VXLAN BGP EVPN fabric, provides ingress route optimization for traffic from the clients to the data center (Refer figure below).

Figure 39: LISP functional roles in a VXLAN BGP EVPN fabric



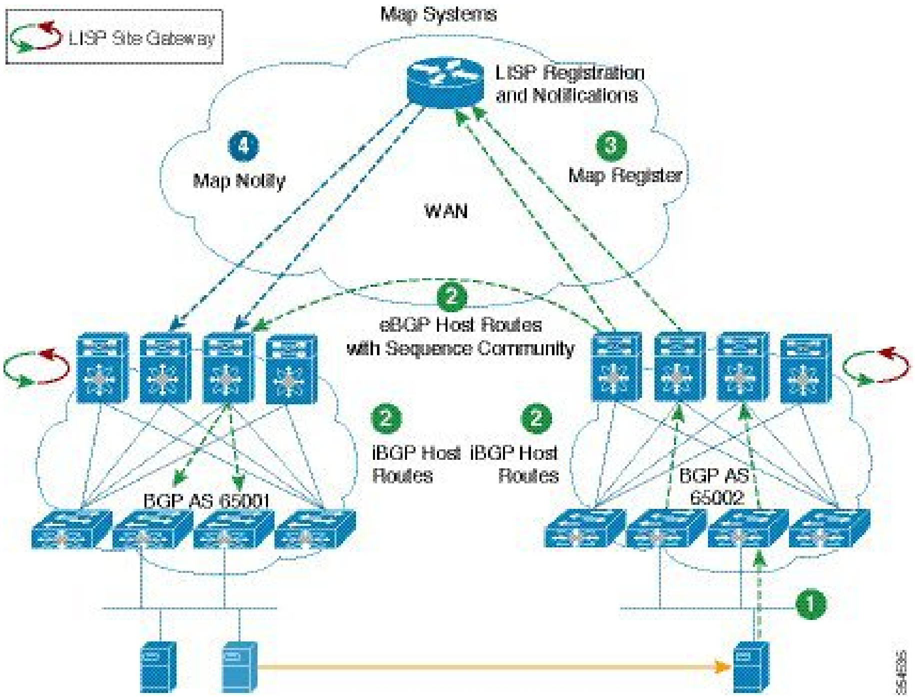
Host Mobility

When the leaf or ToR switch detects a host movement across data centers, it injects that host route into the MP-BGP EVPN control plane with an updated sequence number. The sequence number is a mobility community attribute that represents the state of mobility. It increments every time the server moves from one location to another. This sequence number attribute has to be carried to the original leaf or ToR switch from which the host moved, because it needs to withdraw that particular host route from BGP. The host route withdrawal happens only when the leaf or ToR switch receives a route with an updated sequence number. LISP currently cannot carry the mobility community attribute across the data center through the WAN.

To help LISP achieve mobility semantics across VXLAN BGP EVPN fabrics, you need to establish an Exterior BGP (eGBP) relationship between the data centers. This eBGP relationship is used to carry the mobility community attribute in BGP EVPN across the data center sites for the state reachability information. Details are given below.

Note For traffic between hosts that are in different datacenters but within the same subnet , a Layer-2 tunnel (using a Layer-2 DCI like OTV) is required between the two datacenter fabric pods.

Figure 40: Host Mobility across VXLAN BGP EVPN fabrics



The numbered sequence in the above image is elaborated below:

Step 1—The end system or server, after moving to a new location, sends a DHCP and ARP packet to join the new network.

Step 2—The leaf or ToR switch detects the new host and redistributes the IP address and MAC reachability information in the MP-BGP EVPN control plane with an updated sequence number. This sequence number attribute is carried across the data centers using an eBGP relationship between AS 65001 and 65002. When the original leaf or ToR switch receives the route information with an an updated sequence number, it withdraws its original route from BGP.

When the host first comes online (before moving across data centers), the sequence number attribute will be 0. This value indicates that this was the first time that the host is coming online in any data center (refer *Host mobility with sequence number 0* image below).

Table 3: Host mobility with sequence number 0

MAC	IP	VNI	Next-Hop	Encap	Seq
MAC1	IPHOST1	5000	VTEP L1	VXLAN	0

After the host moves from one location to another, the sequence number is updated to 1, which triggers the route update through the eBGP connection and the route withdrawal from the original leaf or ToR switch (refer *Host mobility with sequence number 1* image below).

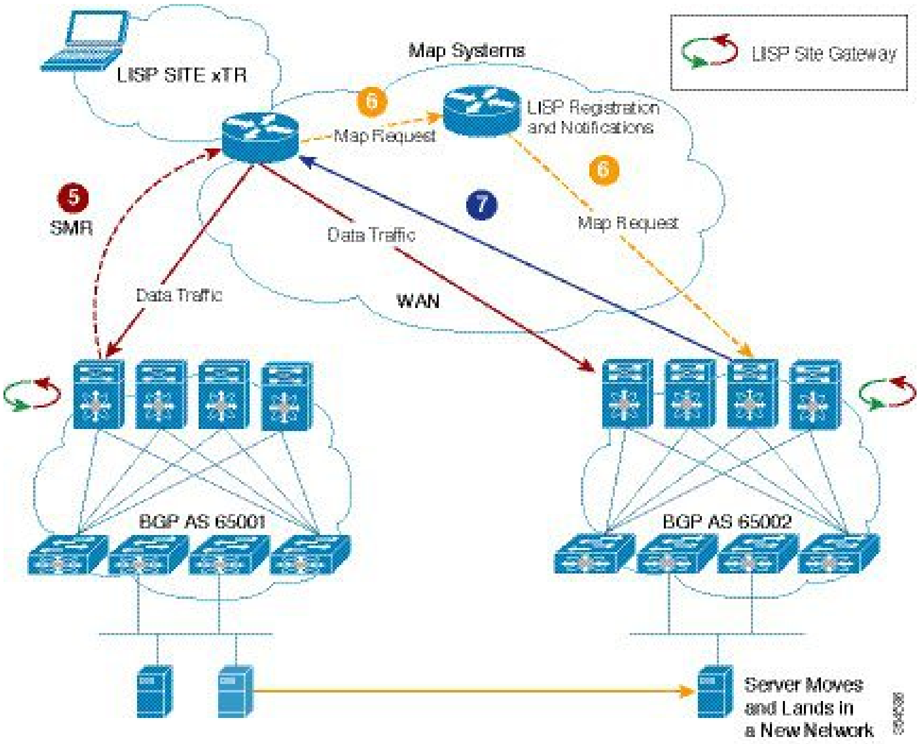
Table 4: Host mobility with sequence number 1

MAC	IP	VNI	Next-Hop	Encap	Seq
MAC1	IPHOST1	5000	VTEP L3	VXLAN	1

Step 3—When the LISP site gateway (also running MP-BGP EVPN in the fabric) detects this new host, it sends a map-register message to the map-system database to register the new IP address in its own data center (BGP AS 65002).

Step 4—When the map system receives the map-register message from BGP, AS 65002 sends a map-notify message to the old LISP site gateways, notifying them that the host has moved from their data center. This message helps ensure that the LISP site gateways install a Null 0 route for that prefix in their routing tables. This Null 0 prefix indicates that the host is in a location remote to that data center.

Figure 41: LISP Map System updates



Step 5—When the clients in the remote branch sites try to send traffic to the LISP site gateways at which the host was present (BGP AS 65001) before the mobility event, the site gateways see that the host is reachable through a Null 0 route. This event triggers a solicit-map request (SMR) from the site gateways to the LISP-enabled router in the branch site asking it to update its database.

Step 6—The branch router then sends a map request to the mapping system asking for the new location of the host. This request is relayed to the LISP site gateways to which the host has moved (BGP AS 65002).

Step 7—The LISP site gateways in BGP AS 65002 unicast a map reply to the LISP-enabled branch router asking it to update its database with the new location.

Now data traffic starts to flow to the correct data center (BGP AS 65002).

Configuration for the VXLAN BGP EVPN – LISP scenario

Hardware and software versions used in the configuration example are given below:

Table 5: Hardware and software versions

Functional Role	Hardware Platform	Software Version
Border spine and border leaf switch	Cisco Nexus 7000 and 7700 Series switches with F3/M3 line card	For the F3 line card, support was introduced in the Cisco NX-OS Software Release 7.2. For the M3 line card, support was introduced in the 8.2(1) release.
Map server and map resolver	Cisco ASR 1000 Series Aggregation Services Routers	Cisco IOS XE Software Release 3.13.2

Border spine switch configuration in Data Center 1 (BGP AS 65001)

This section summarizes the steps for configuring LISP for a hand-off from VXLAN on the border spine or border leaf switch.

Note

Important—This document contains LISP related configurations on the border spine/leaf switch. You should also enable VXLAN BGP EVPN configurations on this switch. For VXLAN BGP EVPN configuration details, refer the *Forwarding Configurations* chapter, *Cisco Nexus 7000 Series switch configuration* section.

Enable the LISP feature and LISP encapsulation/decapsulation functionality

(config) #

```
feature lisp
ip lisp itr
ip lisp etr
```

Configure LISP map server and map resolver reachability

(config) #

```
ip lisp itr map-resolver 10.8.12.9
ip lisp etr map-server 10.8.12.9 key 0 123456789
```

The above commands configure the LISP map resolver address for the LISP ITR functionality, and the LISP map server address (along with a shared key) for the LISP ETR functionality.

Configure the LISP hand-off for tenant VRF instances—The following example shows a two-tenant VRF instance configuration.

(config) #

```
vrf context evpn-tenant-1
  lisp instance-id 10
  ip lisp locator-vrf default
  lisp dynamic-eid lisp-subnets
    ip lisp database-mapping 10.0.0.0/8 10.8.12.10 priority 1 weight 1
```

```
register-route-notifications tag 65001
```

- The above commands creates a LISP instance ID (which needs to be the same across data centers) and defines the RLOC VRF to use (The mapping database is reachable through that RLOC VRF).
- The tag in the route notification (65001) has to match the BGP Autonomous Systems Number that datacenter 1 and the spine switch belong to.
- The **database-mapping** command configures an EID-to-RLOC mapping relationship and its associated traffic policy for the LISP site.
- The **register-route-notifications** command triggers mobility registration on reception of host routes with the tag configured.

A sample configuration of the second tenant VRF is given below

(config) #

```
vrf context evpn-tenant-2
   lisp instance-id 20
   ip lisp locator-vrf default
   lisp dynamic-eid lisp-subnets
     ip lisp database-mapping 10.21.0.0/8 10.8.12.10 priority 1 weight 1
     register-route-notifications tag 65001
```

If you need to configure additional EID (IP address) subnets to map to the VRF instance, then you will have to create another dynamic EID subnet name. A sample configuration is given below.

(config) #

```
vrf context evpn-tenant-2
    lisp dynamic-eid lisp-subnets-1
        ip lisp database-mapping 209.165.200.225/24 10.0.0.2 priority 1 weight 1
        register-route-notifications tag 650001
```

The LISP instance ID provides a means of maintaining unique address spaces in the control and data plane. Instance IDs are numerical tags defined in the LISP canonical address format (LCAF). The instance ID has been added to LISP to support virtualization.

When multiple organizations within a LISP site are using private addresses as EID prefixes, their address spaces must remain segregated to prevent address duplication. An instance ID in the address encoding can be used to create multiple segmented VPNs within a LISP site at which you want to keep using EID-prefix based subnets. The LISP instance ID is currently supported in LISP ingress tunnel routers and egress tunnel routers (ITRs and ETRs), map server (MS), and map resolver (MR).

The LISP locator VRF is used to associate a VRF table through which the routing locator address space is reachable with a router LISP instantiation.

Border Leaf Configuration in Data Center 2 (BGP AS 65002)

Configuration of the border leaf switch is similar to that of the border spine switch. A consolidated configuration is given below.

(config) #

```
feature lisp
ip lisp itr
ip lisp etr
ip lisp itr map-resolver 10.8.2.45
ip lisp etr map-server 10.8.2.45 key 0 123456789
vrf context evpn-tenant-1
   lisp instance-id 10
   ip lisp locator-vrf default
   lisp dynamic-eid lisp-subnets
     ip lisp database-mapping 10.0.0.0/8 10.8.2.46 priority 1 weight 50
     register-route-notifications tag 65002
vrf context evpn-tenant-2
   lisp instance-id 20
   ip lisp locator-vrf default
   lisp dynamic-eid lisp-subnets
     ip lisp database-mapping 10.21.0.0/8 10.8.2.46 priority 1 weight 50
     register-route-notifications tag 65002
```

The tag in the route notification (65002) has to match the BGP Autonomous Systems Number that datacenter 2 and the border leaf switch belong to.

LISP Map-System Database Configuration

Configure the map server and map resolver on the switch. The map server and map resolver can be either on the same device or multiple devices. The scenario here uses an ASR 1000 Series router as the map server and map resolver.

(config) #

```
router lisp
lisp site DC
 authentication-key shared-key
    eid-prefix instance-id 10 10.0.0.0/8 accept-more-specifics
    eid-prefix instance-id 10 10.21.0.0/8 accept-more-specifics
```

The above commands defines the data center administrative scope, and maps the EID prefixes from the data center sites to the LISP mapping system.

(config) #

```
lisp site branch
  authentication-key shared-key
   eid-prefix instance-id 10 10.12.0.0/8 accept-more-specifics
```

The above commands define the branch location administrative scope.

Configure the branch site

(config) #

```
router lisp
   eid-table default instance-id 10
       database-mapping 10.1.0.0/24 10.100.0.1 priority 1 weight 50

router lisp
    ipv4 itr map-resolver 10.9.9.9
    ipv4 etr map-server 10.9.9.9 key s3cr3t-k3y
```

The above commands configure the LISP Map resolver and Map server addresses.

(config) #

```
router lisp
    ipv4 itr
    ipv4 etr
```

The above commands configure the device as a LISP ITR and ETR.

Verification

To check for the EID (host IP address) learnt on the LISP site gateway on a Cisco Nexus 7000 Series or 7700 platform, use the command shown here.

```
N7700-Border-Spine# show lisp dynamic-eid summary vrf evpn-tenant-1

LISP Dynamic EID Summary for VRF " evpn-tenant-1"
* = Dyn-EID learned by site-based Map-Notify
! = Dyn-EID learned by routing protocol
^ = Dyn-EID learned by EID-Notify

Dyn-EID-Name  Dynamic-EID Interface Uptime   Last-Packet Pending-Ping-Count

lisp-subnets  !10.1.1.12  Eth4/1    06:50:21 00:12:12    0
lisp-subnets  !10.1.1.13  Eth4/2    03:20:01 00:10:12    0
```

In the above sample output, **lisp-subnets** refers to the EID subnet name, **Dynamic-EID** column refers to the End host IP addresses (EID), and **Interface** refers to the interface connecting to the leaf switches.

To check for LISP map-cache entries on the map server, use the command shown below:

```
Map-Server# show lisp site

LISP Site Registration Information
* = Some locators are down or unreachable

Site-Name Last-Register Up  Who-Last-Registered Inst-ID EID-Prefix

DC        Never         No  ---                 10      10.0.0.0/8
          00:00:50      Yes 10.8.2.46           10      10.1.1.12/32
          00:00:50      Yes 10.8.2.46           10      10.1.1.13/32
          00:00:40      Yes 10.8.12.10          10      10.1.1.15/32
          00:00:40      Yes 10.8.12.10          10      10.1.1.16/32
          Never         No  ---                 10      10.21.0.0/8
```

In the above example, the **Who Last Registered** column refers to the LISP site gateway/RLOC.

CHAPTER 9

Unicast Forwarding

Unicast

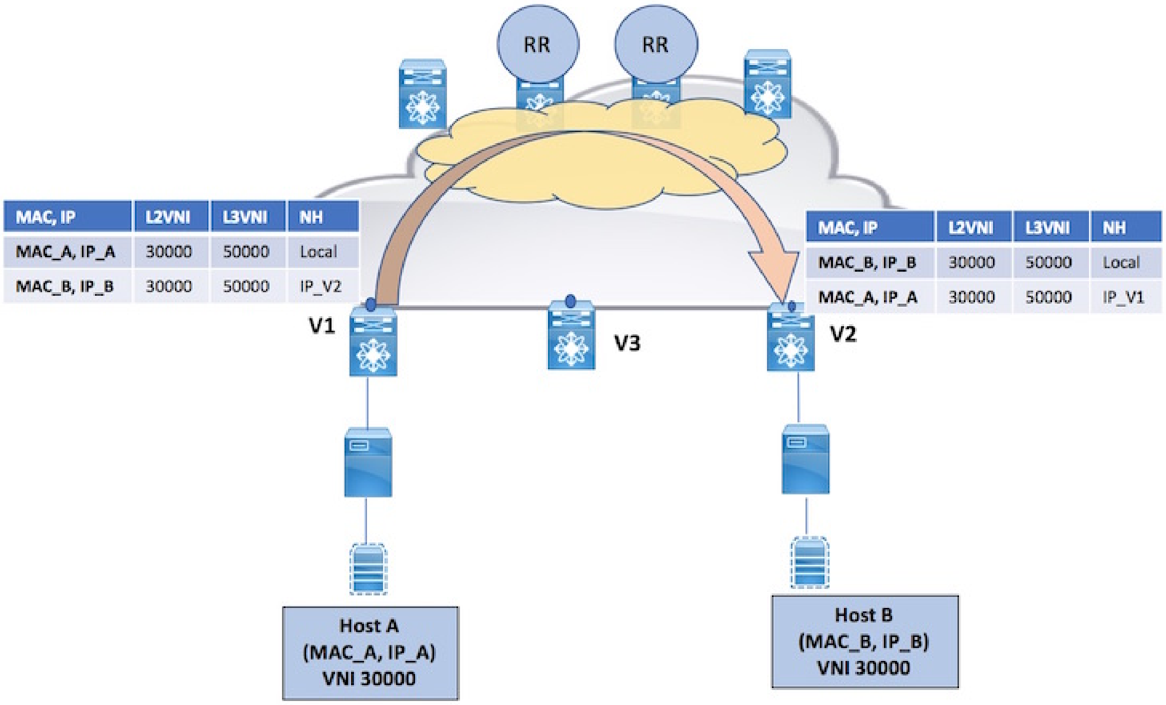
Unicast Forwarding Flows—Overview

Intra and inter subnet forwarding are the possible unicast forwarding flows in the VXLAN BGP EVPN fabric, between leaf/ToR switch VTEPs. They are explained in the subsequent sections.

Intra Subnet Forwarding (Bridging)

Traffic between end hosts within a Layer-2 virtual network is bridged. The reachability information for the 2 end hosts (subnet route, MAC address and IP address) is sent through the MP-BGP EVPN control plane, and the ARP tables in the source and target VTEPs contain this reachability information.

Figure 42: Intra subnet forwarding (Bridging)



MAC, IP	L2VNI	L3VNI	NH
MAC_A, IP_A	30000	50000	Local
MAC_B, IP_B	30000	50000	IP_V2

MAC, IP	L2VNI	L3VNI	NH
MAC_B, IP_B	30000	50000	Local
MAC_A, IP_A	30000	50000	IP_V1

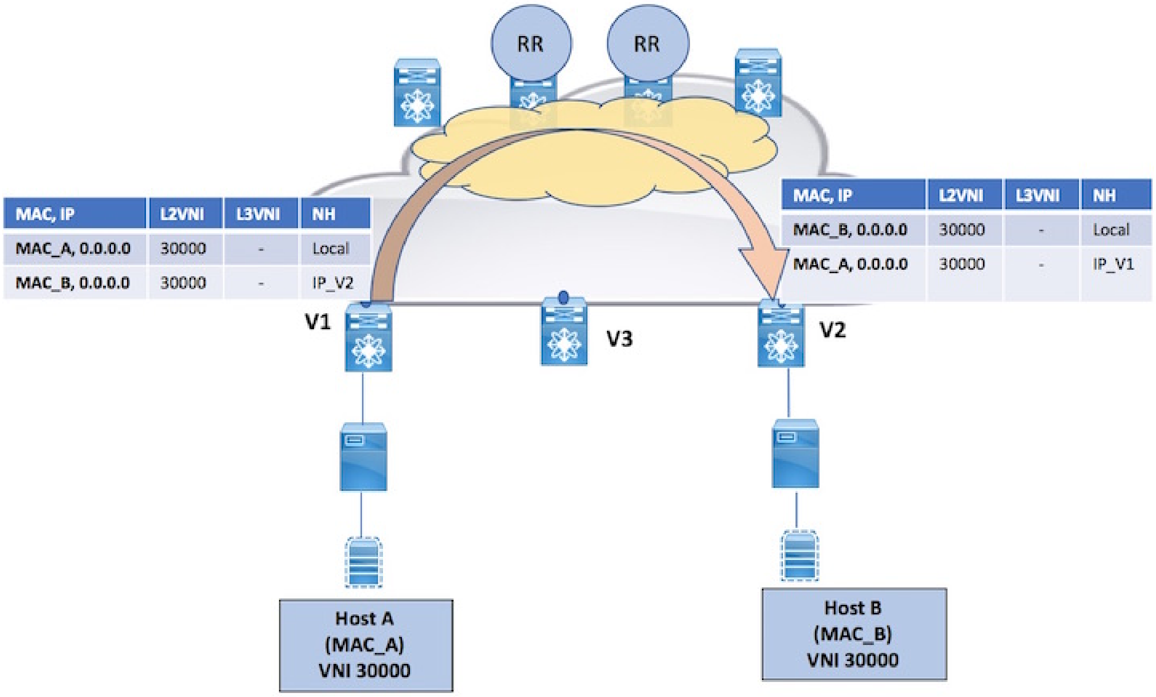
In the above image, Host A on V1 and Host B on V2 are in the same Layer-2 virtual network, represented by VNI 30000. When Host A sends traffic to Host B, the traffic is bridged from V1 to V2, through a spine switch.

Intra Subnet Non-IP Forwarding (Bridging)

This is a use case where end hosts only have a MAC address and no assigned IP address. Some assumptions for non-IP forwarding (Bridging) between 2 known end hosts are given below:

- End hosts are known to the switch VTEPs and MAC address tables are populated.
- Hosts are known to the MP-BGP EVPN control plane and the MAC addresses are populated within the control plane.
- The communication between the end hosts is on the IP layer.

Figure 43: Non-IP forwarding (Bridging)



MAC, IP	L2VNI	L3VNI	NH
MAC_A, 0.0.0.0	30000	-	Local
MAC_B, 0.0.0.0	30000	-	IP_V2

MAC, IP	L2VNI	L3VNI	NH
MAC_B, 0.0.0.0	30000	-	Local
MAC_A, 0.0.0.0	30000	-	IP_V1

Non-IP forwarding (Bridging)

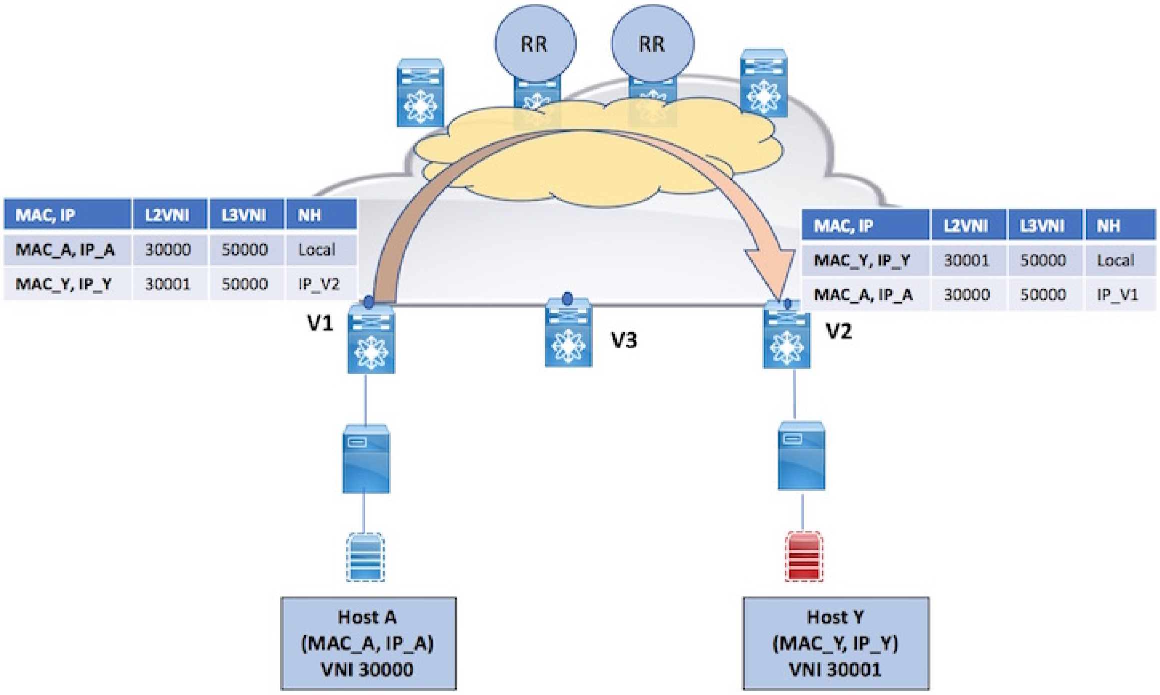
Host A and Host B belong to the same Layer-2 virtual network (with VNI 30000). When Host A sends traffic to Host B, the frame that reaches V1 only contains source and destination MAC addresses. It does not contain IP addresses and Layer-3 VNI information, unlike in the IP bridging scenario where the source and destination end hosts' IP address is updated in the inner frame.

V1 encapsulates this frame (as any incoming frame/packet) in a VXLAN packet and the packet is bridged from V1 to V2, through a spine switch. V2 decapsulates the VXLAN packet and sends the inner frame to Host B.

Inter-Subnet Forwarding (Routing)

Traffic between end hosts of different Layer-2 virtual networks is routed.

Figure 44: Routing across Layer-2 virtual networks



MAC, IP	L2VNI	L3VNI	NH
MAC_A, IP_A	30000	50000	Local
MAC_Y, IP_Y	30001	50000	IP_V2

MAC, IP	L2VNI	L3VNI	NH
MAC_Y, IP_Y	30001	50000	Local
MAC_A, IP_A	30000	50000	IP_V1

Some notes for the image is given below:

- Host A (attached to V1) belongs to the Layer-2 virtual network with VNI 30000, and Host Y (attached to V2) to Layer-2 VNI 30001. The subnets for the 2 networks host different IP address ranges. So, traffic between them is routed.

- The reachability information for the 2 end hosts i.e. the L3 VNI, MAC and IP addresses along with subnet address of the SVI has been sent through the MP-BGP EVPN control plane. ARP (the ARP suppression feature should be enabled for this) and forwarding tables of the source and target switch VTEPs are populated with the end hosts' reachability information.

- *Host A to V1*—When Host A sends traffic to V1, the Destination MAC (DMAC) of the packet is encapsulated with the MAC address of the (distributed IP anycast) gateway.

- *V1 to V2*—V1 does a look up, notes the VTEP that Host Y is attached to, and checks the VRF (and associated L3 VRF VNI). V1 VXLAN encapsulates the traffic sent by Host A and sends it towards V2.

  The routed traffic from V1 to V2 logically traverses through the Layer-3 VRF VNI 50000. Practically, the traffic traverses through the underlay.

Note When a packet is bridged, the target end host's MAC address is entered in the DMAC field of the inner frame. When a packet is routed, the default gateway's MAC address is entered in the DMAC field of the inner frame. This is because, though VXLAN needs inner SMAC and DMAC fields, for routing purposes, the DMAC should contain the anycast gateway address.

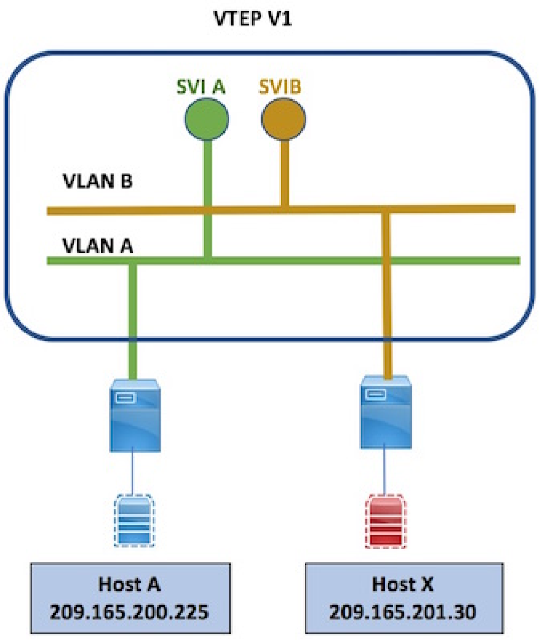
- *V2 to Host Y*—When the packet reaches V2, V2 does a control plane lookup, notes that Host Y's IP address is in VRF A, and then does a MAC table lookup for Host Y. After identifying the port to which Host Y is attached, the packet is sent to Host Y.

Other routing scenarios

Local Routing

Routing between servers belonging to different Layer-2 virtual networks, but attached to the same leaf/ToR switch (VTEP).

Figure 45: Routing across Layer-2 virtual networks on a VTEP



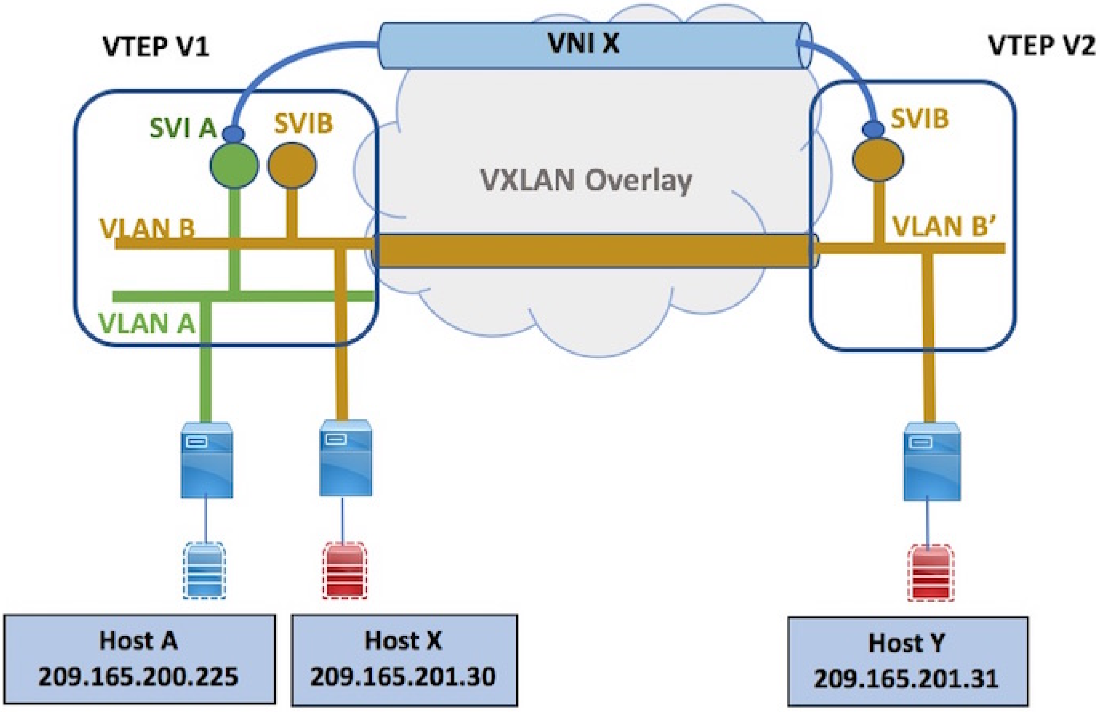
Some notes for the image is given below:

- Host A and Host X belong to different Layer-2 virtual networks, but are attached to the same ToR switch VTEP.
- In this case, the packets are routed through their VLANs (and SVIs). If Host A sends traffic to Host X, the switch does a control plane lookup, identifies that the target is local to the switch, and routes the traffic to Host X.
- The local routing scenario is normal inter VLAN routing. Neither the L2 VNI nor L3 VNI is used for local routing.

Routing to an unknown end host (scenario 1)

Routing to an unknown destination end host (or silent host), an end host that has not ARPed. However, the end host's subnet is known, and has a presence in the source ToR switch VTEP.

Figure 46: Routing to an unknown host (scenario 1)



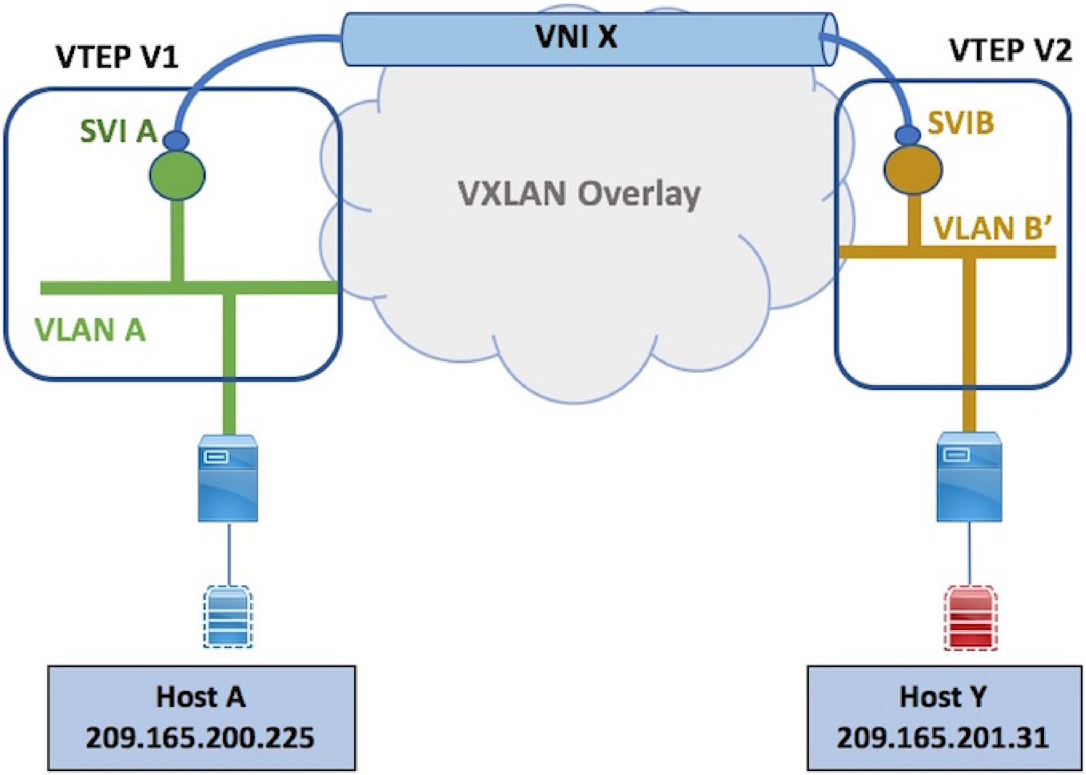
Some notes for the image is given below:

- Host A (on V1) sends traffic to Host Y (on V2). Since the 2 hosts are in different Layer-2 virtual networks, the traffic is routed. Though Host A (and V1) is aware of Host Y's IP address, V1 does not know where Host Y is located.
- V1, however, knows the subnet to which Host Y belongs, and tries to reach Host Y with this information. Since V1 has Host X attached to it, and Host X has the same subnet as that of Host Y, local routing is done within the VTEP to Host X. This is the reason for advertising local subnets on a ToR switch VTEP.
- V1 sends an ARP request for Host Y's IP address. This is only sent to those switch VTEPs that have end hosts for the L2VNI (including the VTEP that has Host Y). The ARP request's source is the anycast gateway MAC and IP address.
- In response to the ARP request, since Host Y is attached to V2, Host Y's response is sent to V2, and the response is populated in the control plane, thereby ensuring that reachability information for Host Y is made available.
- Subsequently, when Host A communicates to Host Y, the traffic is sent from V1 through the Layer-3 VNI (VNI X) to V2, and to Host Y.

Routing to an unknown end host (scenario 2)

Routing to an unknown destination. However, the end host's subnet is known, but the subnet does not have any presence in the source ToR switch VTEP.

Figure 47: Routing to an unknown host (scenario 2)



Some notes for the image is given below:

- Host A (on V1) sends traffic to Host Y (on V2). Though Host A (and V1) is aware of Host Y's IP address, V1 does not know where Host Y is located. Also, V1 does not have any end host that belongs to the same Layer-2 virtual network as that of Host Y.
- V1 knows Host Y's IP address and the subnet to which it belongs (which V1 has learnt previously through the control plane) and sends an ARP request towards the L2VNI on one of the VTEPs advertising the subnet. The ARP request's response ensures Host Y's reachability information is sent in the control plane. The direct route of the SVI is redistributed into the VXLAN BGP EVPN control plane, thereby ensuring that V1 knows where Host Y resides.
- When it is found out that Host Y's subnet is behind VTEP2, Host Y's reachability information is updated on V1. Now, Host Y is reachable across the fabric.

IPv6 address handling in the VXLAN BGP EVPN fabric

Some pointers are given below:

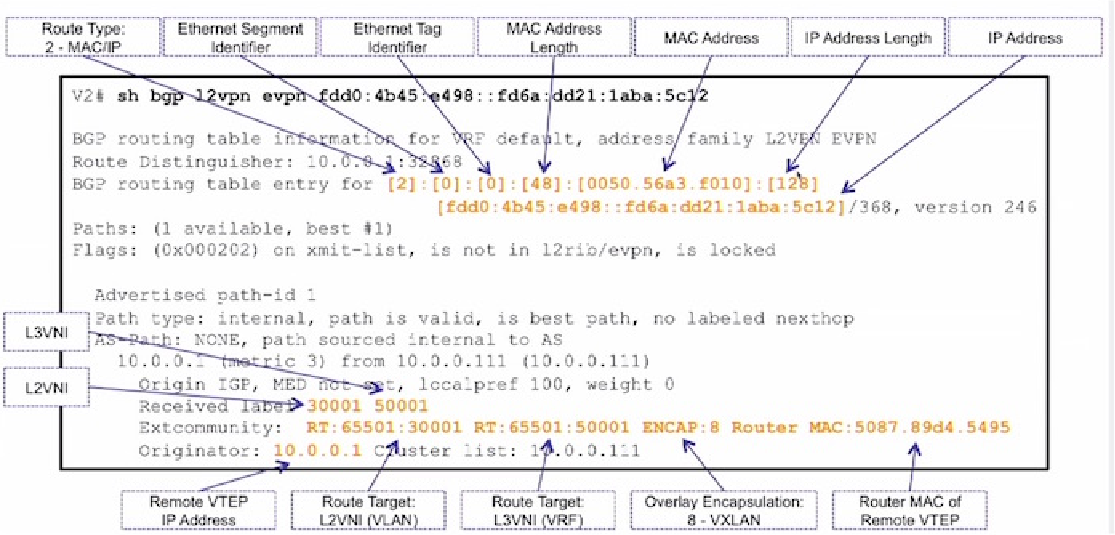
- IPv6 addresses are supported in the fabric overlay, but not in the fabric underlay. So, end hosts can have IPv6 addresses, but the underlay can only contain IPv4 addresses.

- The semantics for IPv4 and IPv6 addressing in the overlay are conceptually the same, with differences that ARP and Neighbor Discovery (ND) have.
- Address resolution - ARP (IPv4) and ND (IPv6). IPv4 addresses consume less system resources on switching hardware tables compared to IPv6 addresses due to the length of the addresses.
- Layer-3 (IPv6) information is learned based on ND snooping.
- Subnet routing is also supported for IPv6 prefixes/subnets.

IPv6 Route Types

IPv6 representations of EVPN Route Type 2 (MAC/IP advertisement routes) and Route Type 5 (IP prefix routes) are given below.

Figure 48: IPv6 host route—Route Type 2

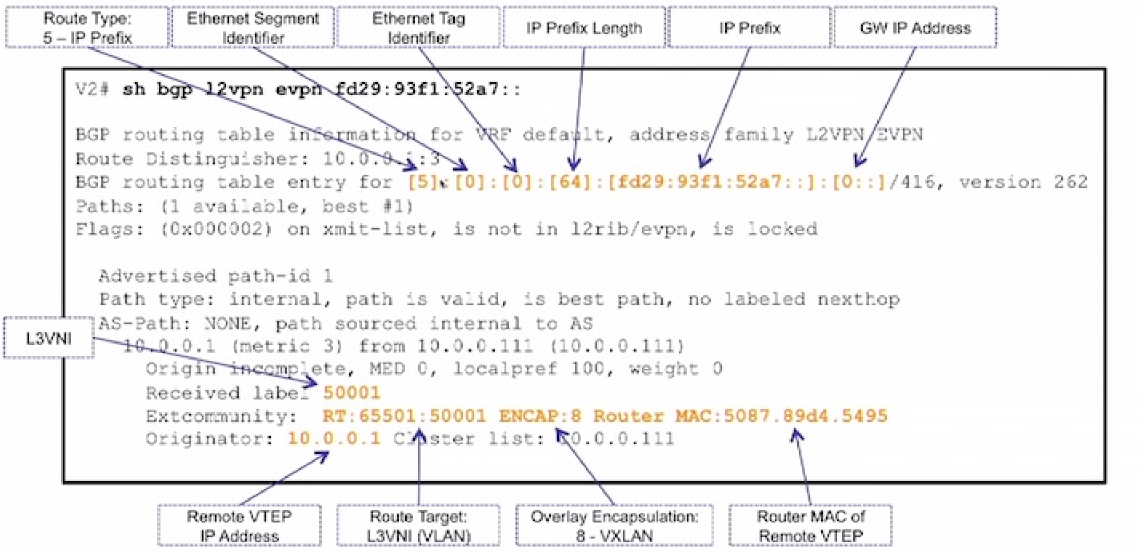


```
V2# sh bgp l2vpn evpn fdd0:4b45:e498::fd6a:dd21:1aba:5c12

BGP routing table information for VRF default, address family L2VPN EVPN
Route Distinguisher: 10.0.0.1:32868
BGP routing table entry for [2]:[0]:[0]:[48]:[0050.56a3.f010]:[128]
                    [fdd0:4b45:e498::fd6a:dd21:1aba:5c12]/368, version 246
Paths: (1 available, best #1)
Flags: (0x000202) on xmit-list, is not in l2rib/evpn, is locked

  Advertised path-id 1
  Path type: internal, path is valid, is best path, no labeled nexthop
  AS-Path: NONE, path sourced internal to AS
    10.0.0.1 (metric 3) from 10.0.0.111 (10.0.0.111)
      Origin IGP, MED not set, localpref 100, weight 0
      Received label 30001 50001
      Extcommunity:  RT:65501:30001 RT:65501:50001 ENCAP:8 Router MAC:5087.89d4.5495
      Originator: 10.0.0.1 Cluster list: 10.0.0.111
```

Some notes for the image are given below:

- The fields in the image are self-explanatory.
- The IP Address Length field displays 128, indicating it's an IPv6 address. The IP Address field displays the IPv6 address. The IPv6 address is being distributed in the control plane.

Figure 49: IPv6 prefix route—Route Type 5



```
V2# sh bgp l2vpn evpn fd29:93f1:52a7::

BGP routing table information for VRF default, address family L2VPN EVPN
Route Distinguisher: 10.0.0.1:3
BGP routing table entry for [5]:[0]:[0]:[64]:[fd29:93f1:52a7::]:[0::]/416, version 262
Paths: (1 available, best #1)
Flags: (0x000002) on xmit-list, is not in l2rib/evpn, is locked

  Advertised path-id 1
  Path type: internal, path is valid, is best path, no labeled nexthop
  AS-Path: NONE, path sourced internal to AS
    10.0.0.1 (metric 3) from 10.0.0.111 (10.0.0.111)
      Origin incomplete, MED 0, localpref 100, weight 0
      Received label 50001
      Extcommunity:  RT:65501:50001 ENCAP:8 Router MAC:5087.89d4.5495
      Originator: 10.0.0.1 Cluster list: 10.0.0.111
```

Some notes for the image are given below:

- The fields in the image are self-explanatory.
- The IP Prefix Length, IP Prefix, and GW IP Address fields denote IPv6 addressing semantics.

CHAPTER 10

Layer-2 Multicast Forwarding

Multicast

Overview

Layer-2 multicast traffic is typically transported in the overlay. In the VXLAN BGP EVPN fabric, Layer-2 multicast refers to the use of IP multicast technology for transporting multicast traffic in a Layer-2 switching environment.

Layer-2 multicast follows the same semantic as general Broadcast, Unknown Unicast, and Multicast (BUM) traffic. On Cisco Nexus 7000 Series and Cisco Nexus 9000 Series switches, IGMP snooping is used to facilitate Layer-2 multicast forwarding in the fabric.

This chapter only explains Layer-2 multicast. Multicast routing, used for transporting Layer-3 multicast traffic, is explained in the *Multicast Routing in the VXLAN Underlay* section, *IP Fabric Underlay* chapter.

Layer-2 Multicast Forwarding

Some pointers are given below:

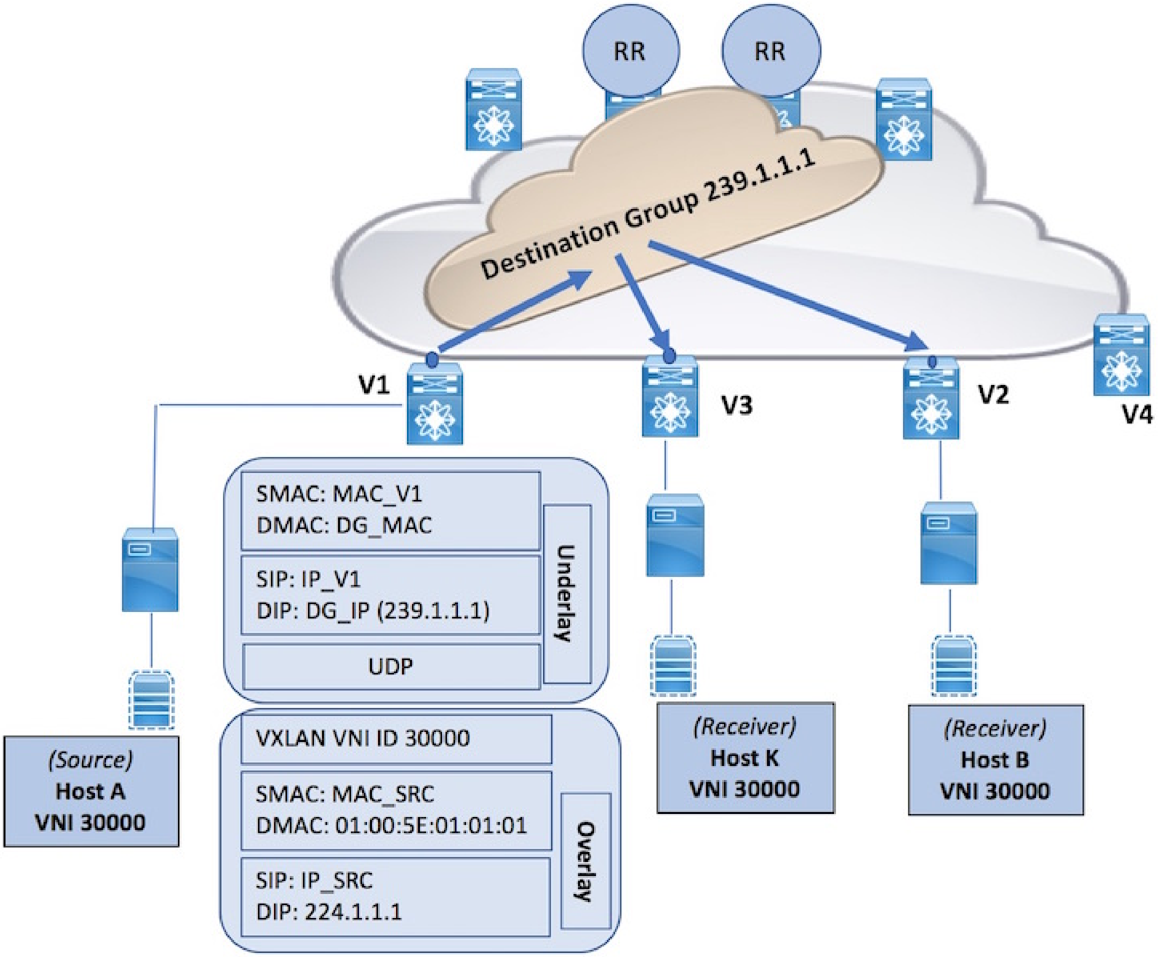
- *Multicast packets are sent only to participating leaf/ToR switch VTEPs*—A Layer-2 virtual network is tied to a unique multicast group. The L2 VNI-to-multicast-group configuration should be enabled only on those switches that have servers of this network connected. That way, only those leaf/ToR switches that have attached servers for the L2 VNI will participate in the multicast group. This will avoid unneeded traffic flow across the fabric.

- You should configure the same Layer-2 VNI to multicast group mapping on all VTEPs where the Layer-2 VNI has presence in.

- The multicast group is used for sending multi destination traffic, like broadcast (ARP requests), unknown unicast, and multicast traffic replication.

Note If Layer-2 multicast packets are to be sent to receivers in another data center, then a Layer-2 Datacenter Interconnect (DCI) technology like OTV is used to connect the two data centers and send the information.

Figure 50: Layer-2 Multicast Forwarding



- In the above example, let us say Host A, attached to V1, belongs to L2 virtual network 30000. With the below sample configuration, VNI 30000 is tied to the underlay multicast group or Destination Group 239.1.1.1.

  (config) #

  ```
  interface nve 1
    member vni 30000
      mcast-group 239.1.1.1
  ```

  The Layer-2 virtual network has presence on V2 and V3, and the above configuration is applied similarly on V2 and V3 too.

- When A sends multicast traffic, V1 receives it and sends it to V2 and V3, and all other leaf/ToR switches that participate in VNI 30000.

Layer-2 multicast traffic flow example

Some details are given below for better understanding:

- A sends multicast traffic to its attached ToR/leaf switch V1. The DMAC and DIP of this packet/frame contains the server side multicast MAC and IP address (224.1.1.1).
- V1 knows that Host A, the source, belongs to L2VNI 30000. V1 VXLAN encapsulates the multicast packet and sends a single copy of the multicast packet upstream. Within the spine switch, the traffic to V2 and V3 is replicated efficiently (single packet per neighbor). This only happens as V2 and V3 are subscribed to the underlay multicast group (Destination Group) 239.1.1.1.

  Note that, in a VXLAN multicast packet, the DMAC and DIP of the outer header contains the MAC and IP address of the destination multicast group
- Upon receiving, V2 and V3 decapsulate the VXLAN packets, check for intended multicast recipients, and send the original frame/packet to attached servers that are subscribed to the server side multicast group 224.1.1.1.

Note More than one Layer-2 VNI can be associated with the same underlay multicast group (Destination Group). In case multiple virtual networks (say, VNIs 30000 and 30001) are assigned to the same group 239.1.1.1, and there are VTEPs that serve only end hosts of 30000 and not 30001, those VTEPs will still receive multicast traffic for 30001's end hosts, but drop those packets if the Layer-2 segment is not instantiated. This is because all VTEPs that subscribe to a multicast group need to receive traffic for that group.

Multicast Communication in virtual Port Channel (vPC) Scenarios

A common, virtual IP address (VIP) is assigned to represent both the vPC peers for receiving and sending multicast (and unicast) packets.

- When a multicast source is behind a vPC complex, the VIP will be used as the outer source IP address of the VXLAN multicast packet.
- When a multicast receiver is behind a vPC complex, the receiver is announced by the VIP that represents the vPC pair. Multicast traffic intended for this receiver will only be sent to one of the two vPC nodes, since only one node will be active in the underlay multicast tree.

Layer-2 Multicast Configuration in the VXLAN BGP EVPN fabric

The following is a sample Layer-2 multicast configuration for two Layer-2 virtual networks with VNIs 30000 and 30001.

Important This is a partial configuration in the overall VXLAN BGP EVPN fabric configuration, and should not be done in isolation. For complete configuration, refer the *Forwarding Configurations* chapter, *Cisco Nexus 5600 Series switch configuration* and *Cisco Nexus 7000 Series switch configuration* sections. These sections contain complete unicast and multicast configurations.

(config) #

```
feature pim
interface nve 1
   member vni 30000
      mcast-group 239.1.1.1
```

(config) #

```
interface nve 1
   member vni 30001
      mcast-group 239.1.1.2
```

Note Ensure that you retain the same Layer-2 VNI to multicast group mapping on all VTEPs where the Layer-2 VNI has presence in.

CHAPTER 11

Multi-tenancy

Multi-tenancy

Multi-tenancy Overview

Multi-tenancy is a mode of operation where multiple independent instances (Layer-3 VRFs, Layer-2 VLANs) of a tenant (business entity, user group, applications, or security) operate in a shared environment (VXLAN BGP EVPN fabric), while ensuring logical segmentation between the instances. The tenant instances such as VRF and VLANs are logically isolated but physically operate on the same fabric.

Layer-3 and Layer-2 VNIs

In a VXLAN BGP EVPN fabric, a Layer-3 Virtual Network Identifier (VNI) identifies each tenant at the Layer-3 level and is associated with a unique tenant VRF.

As a pendant to Layer-3, Layer-2 virtual networks (VLANs) can carry a unique Layer-2 Virtual Network Identifier (VNI) in the fabric. Separate Layer-2 and Layer-3 networks can be created to achieve Layer-2 and Layer-3 segmentation, like for business units, user groups, applications, etc. Typically, a Layer-2 virtual network is associated with a single IP subnet while a VRF can contain multiple Layer-2 networks.

Note **Important**—From a global, VXLAN BGP EVPN fabric perspective, the VNI is the important identifier that is used across the fabric.

Servers belonging to a Layer-2 virtual network can be spread across the fabric, and might be associated with different Top of Rack (ToR)/leaf switches. Communication between servers or end hosts of the same Layer-2 virtual network is typically bridged.

Communication between end hosts belonging to different Layer-2 virtual networks represents Layer-3 communication, and is achieved through routing. Routed traffic traversing through the fabric logically traverses through the Layer-3 VNI or VRF VNI. The Layer-3 virtual network is similarly spread across different TOR/leaf switches to match the respective Layer-2 virtual networks that require routing.

Across the VXLAN BGP EVPN fabric, the L2 and L3 VNIs are unique and have a global significance. All the Layer-2 virtual networks of a tenant or VRF are associated with a common, unique, Layer-3 VRF VNI.

Note The L2 and L3 VNI use the same VNI field in the VXLAN encapsulation and hence can’t overlap.

Figure 51: Logical representation of server traffic and segregation across the VXLAN BGP EVPN fabric

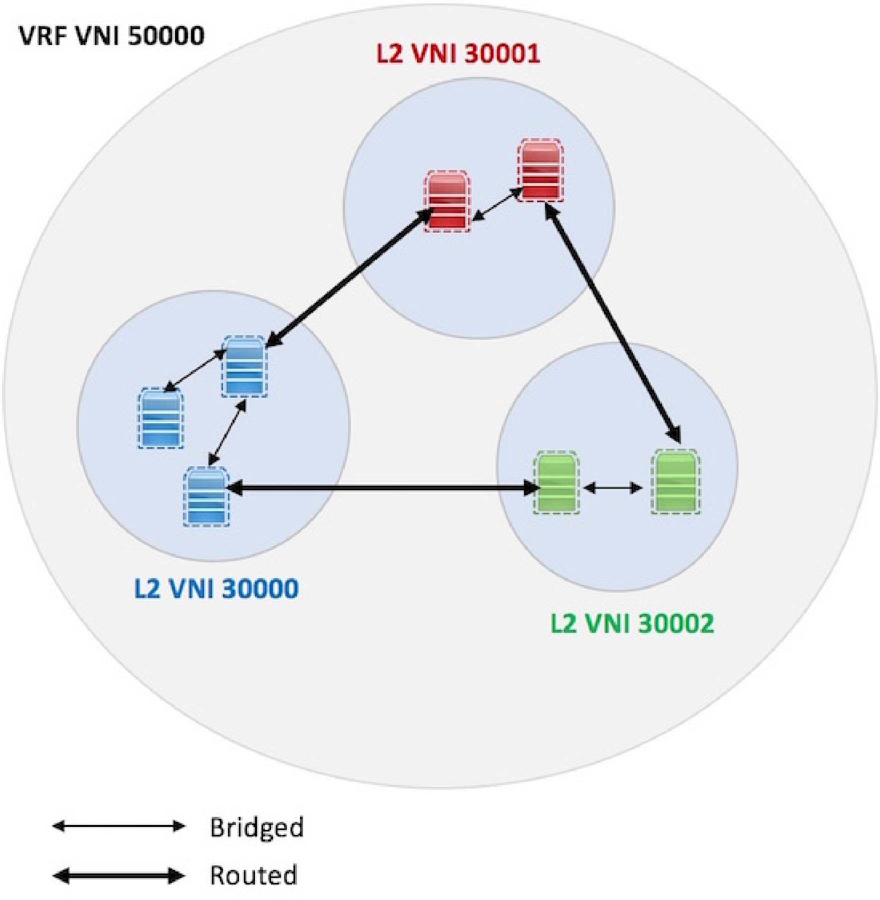
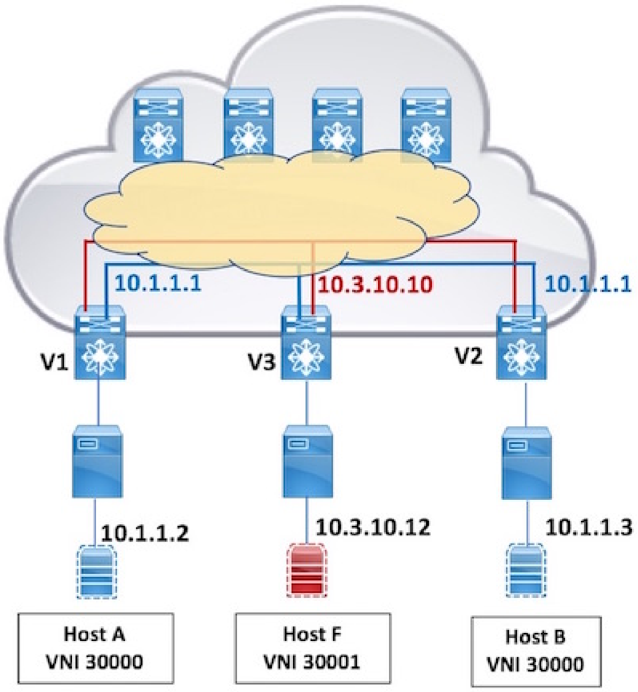


Figure 52: Physical representation of server traffic and segregation across the VXLAN BGP EVPN fabric



In the above sample topology, Host A and Host B belong to the same Layer-2 virtual network (VNI 30000, in blue color), so the traffic between them is bridged. Traffic from Host A to Host F (VNI 30001, in red color) is routed through the VRF VNI (say VNI 50000).

Routing between Layer-2 virtual networks

In the VXLAN BGP EVPN fabric, each Layer-2 virtual network needs to be configured with a first hop gateway approach. This first hop gateway will allow to traverse through the Layer-3 boundary, and send traffic to an end host in another Layer-2 virtual network. Since a Layer-2 virtual network might have presence across the fabric with its end hosts attached to multiple ToRs, the same first hop gateway IP address should be configured on those ToR switches where it has presence (Distributed Anycast Gateway).

In the above example, to route traffic from Host A (Layer-2 VNI 30000) to Host F (Layer-2 VNI 30001), you should configure a first hop gateway IP address (say 10.1.1.1/24) on the attached ToR switch V1. To route Host B traffic to Host F, you should enable the same first hop gateway (10.1.1.1/24) on the attached ToR switch V2. This is because Host A and B belong to the same Layer-2 network (VNI 30000). When using the Distributed Anycast Gateway as a first hop gateway, the IP address as well as the MAC address for the gateway itself will be the same across all ToR switches.

Layer-2 and Layer-3 Multi-tenancy

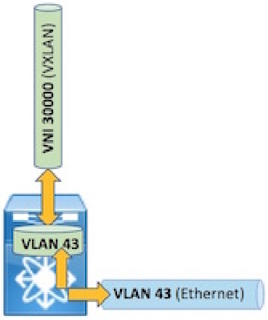
Let us consider bridging (Layer-2 multi-tenancy operation) and routing (Layer-3 multi-tenancy operation) between end hosts across the VXLAN BGP EVPN fabric, from a multi-tenancy perspective. The following sections explain how tenant instances (VLANs and VRFs) are connected across the fabric to send Layer-2 bridged and Layer-3 routed traffic from an end host to another.

For convenience, Cisco Nexus 9000 Series and 7000 Series switch concepts are explained separately.

Layer-2 Multi-tenancy on Cisco Nexus 9000 Series (and Cisco Nexus 5600 Series) Switches

VLAN based multi-tenancy can be implemented on the 9000 and 5600 Series switches. Some pointers are given below:

Figure 53: Layer-2 multi-tenancy—Same parent VLAN and VLAN on the wire



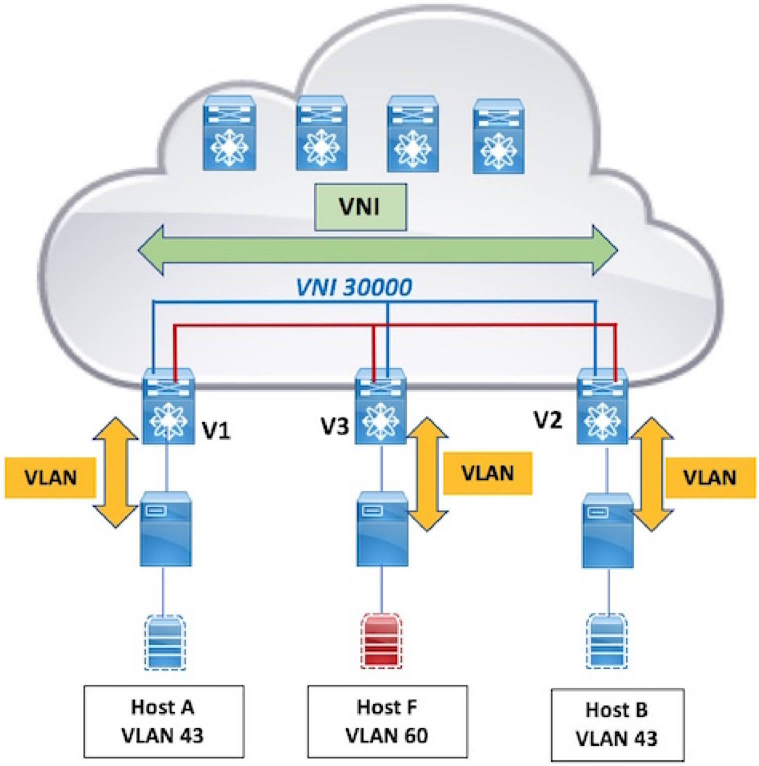
- From a ToR switch's perspective, a Layer-2 virtual network is represented by a VNI on the VXLAN BGP EVPN fabric side (VNI 30000 in the image) and a unique VLAN (43) on the tenant side. The 1:1 mapping between the parent VLAN and the VNI should be configured on the ToR switch.

Note A ToR switch can traditionally only accommodate the 12-bit VLAN namespace. However, the VLAN limitation at the network level is removed due to the introduction of VNIs or segments in the fabric.

- The use case below shows an end-to-end example.

Layer-2 multi-tenancy use case—Same parent VLAN and VLAN on the wire

Figure 54: Same parent VLAN and VLAN on the wire—VXLAN BGP EVPN fabric



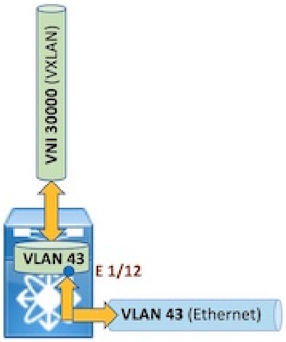
- In the above example, on ToR switch VTEP V1, VLAN 43 is mapped to L2 VNI 30000. All end host ports that have servers of this network should be associated with VLAN 43.
- Host B on V2 belongs to the same Layer-2 virtual network (30000). On ToR switch VTEP V2, Host B is mapped to VLAN 43 and VLAN 43 to VNI 30000.

Note Since VLANs only have ToR/leaf switch significance, different switches can have different VLAN IDs to represent a L2 network. However, it is convenient if you use the same VLAN ID across switches. The L2 VNI binds the Layer-2 virtual network and extends Layer-2 reachability across the fabric.

A sample configuration is given below:

Note **Important**—Configurations in this chapter/white paper are partial configurations in the overall VXLAN BGP EVPN fabric configuration, and should not be done in isolation. For complete configuration, refer the *Forwarding Configurations* chapter, Cisco Nexus 9000, Cisco Nexus 5600 Series switch configuration and Cisco Nexus 7000 Series switch configuration sections.

Figure 55: Same parent VLAN and VLAN on the wire – ToR switch VTEP



Create a VLAN and map it to a L2 VNI

(config) #

```
vlan 43
  vn-segment 30000
```

VLAN 43 and VNI 30000 are mapped with each other.

Configure an access port to enable Layer-2 traffic on the interface

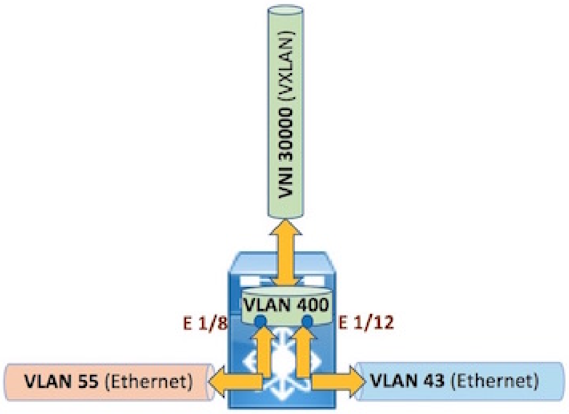
(config) #

```
interface Ethernet 1/12
   switchport mode access
   switchport access vlan 43
```

Layer-2 multi-tenancy use case – VLANs on the wire are mapped to a parent VLAN

Note The manual configuration of VLAN translation is only supported on the Cisco Nexus 9000 Series switches and not on Cisco Nexus 5600 Series switches.

Figure 56: VLAN translation—Multiple VLANs mapped to a parent VLAN



Parent VLAN

In this use case, VNI 30000 represents a Layer-2 virtual network on the fabric side. VLAN 400 is the parent VLAN on the ToR switch that represents the Layer-2 virtual network on the tenant side. VLAN 400 needs to be mapped to VNI 30000 for Layer-2 stitching or extension.

VLANs on the wire

VLANs 55 and 43 are on the wire, and end hosts are mapped to these VLANs. They also represent the Layer-2 virtual network 30000, but only through VLAN 400, the parent VLAN.

A sample configuration is given below:

Create a Parent VLAN and map to a L2 VNI

(config) #

```
vlan 400
  vn-segment 30000
```

Configure a trunk port to enable Layer-2 traffic across VLANs—Ethernet interface 1/8

(config) #

```
interface Ethernet 1/8
  switchport mode trunk
  switchport vlan mapping enable
  switchport vlan mapping  55 400
  switchport trunk allowed vlan 400
```

VLAN 55 is local and significant only to the port on which it is configured (Ethernet 1/8 in this case). VLAN 55 is mapped to parent VLAN 400.

Configure a trunk port to enable Layer-2 traffic across VLANs—Ethernet interface 1/12

(config) #

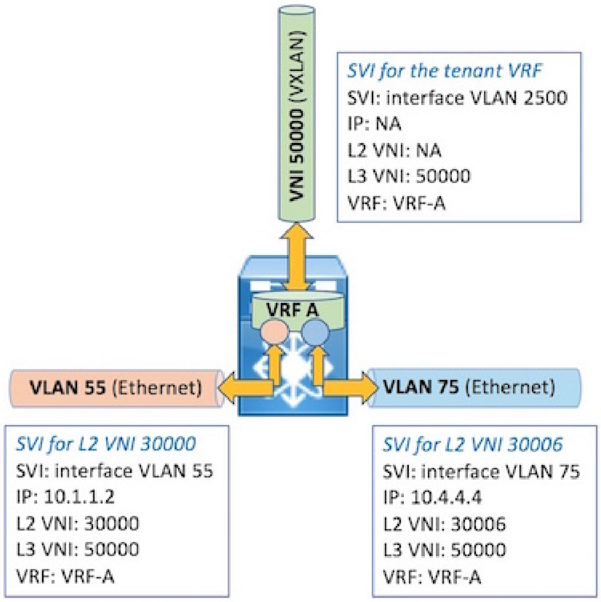
```
interface Ethernet 1/12
  switchport mode trunk
  switchport vlan mapping enable
  switchport vlan mapping 43 400
  switchport trunk allowed vlan 400
```

VLAN 43 is local and significant only to the port on which it is configured (Ethernet 1/12 in this case). VLAN 43 is also mapped to parent VLAN 400

Layer-3 multi-tenancy on Cisco Nexus 9000 Series (and Cisco Nexus 5600 Series) Switches

To route traffic between L2 virtual networks, an L3 virtual interface (called switch virtual interface [SVI]) should be created for each L2 virtual network, on a ToR/leaf switch. Each SVI should be associated with the tenant VRF, thereby adding it to the VRF table. A sample scenario is given below:

Figure 57: Layer-3 multi-tenancy

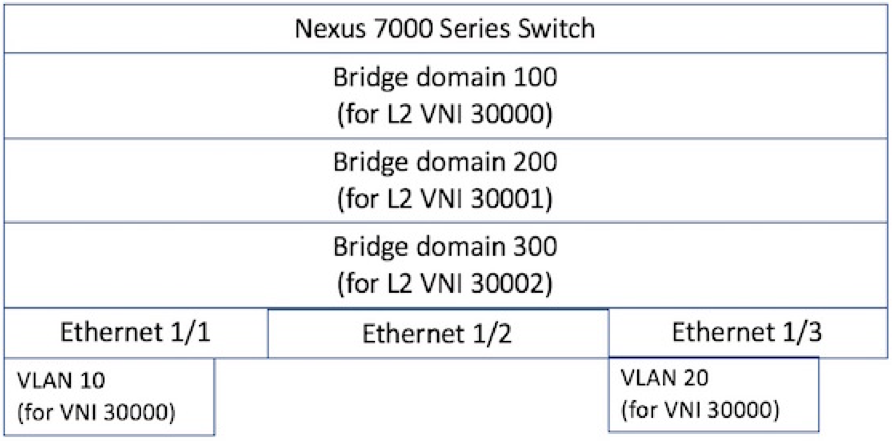


- Tenant VRF A is tied to Layer-3 VNI 50000. On this leaf switch/ToR VTEP, this tenant has the presence of L2 virtual networks 30000 and 30006.
- VLAN 2500 is created for L3 VNI 50000. An SVI, *interface VLAN2500*, is created for the tenant VRF.
- Similarly, SVIs are created for L2 VNI 30000 (VLAN 55), and L2 VNI 30006 (VLAN 75). The SVIs are *interface VLAN 55* and *interface VLAN 75*. An IP address is assigned to each SVI.

Layer-2 multi-tenancy on Cisco Nexus 7000 Series switches

Bridge domain based multi-tenancy can be implemented on the 7000 Series switches.

Figure 58: A depiction of the relationship between bridge domains, VLANs, and ethernet ports



Nexus 7000 Series Switch

Bridge domain 100
(for L2 VNI 30000)

Bridge domain 200
(for L2 VNI 30001)

Bridge domain 300
(for L2 VNI 30002)

Ethernet 1/1

Ethernet 1/2

Ethernet 1/3

VLAN 10
(for VNI 30000)

VLAN 20
(for VNI 30000)

Some pointers about bridge domain based multi-tenancy are given below:

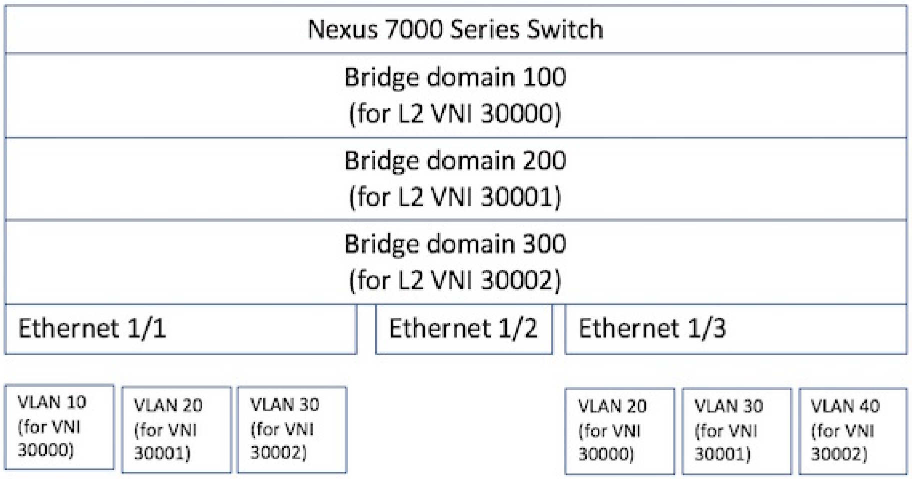
- *Bridge domain*—It is a Layer-2 flood broadcast domain that maps/connects VLANs of a Layer-2 virtual network on the tenant side of a ToR switch to a Layer-2 VNI on the fabric side of the switch. There is a 1:1 mapping between the bridge domain ID and the L2 VNI (one BD for each L2 virtual network). The most common known representation of a bridge domain is a VLAN.

  > **Note** **Important**—Here, VLANs have port level significance. For example, VLAN 10 represents Layer-2 virtual network 30000 on Ethernet 1/1 and VLAN 20 represents it on Ethernet 1/3. So, BD 100 (BD for L2 VNI 30000) bridges L2 end host traffic between Eth 1/1 and Eth 1/3, and from either (VLAN) port to end hosts in the same L2 virtual network across the fabric.

  MAC learning is done on the bridge domain (e.g. BD 100) that maps to VNI 30000.

- *Flexibility*—You can reuse VLAN IDs across ports (see image *Port specific multi-tenancy* below – VLAN 20 ties to VNI 30001 on Eth 1/1 but to VNI 30000 on Eth 1/3), and reuse BD IDs across ToR switches. A ToR switch in this model can accommodate 4K VLANs per 802.1Q trunk port. The VLAN limitation at the network level is removed due to the introduction of VNIs in the fabric.

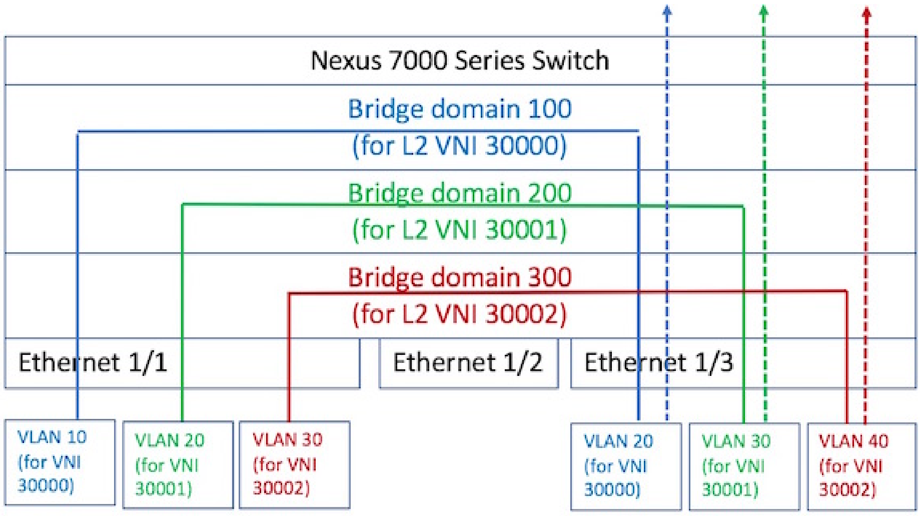
Figure 59: Port specific multi-tenancy



- The L2 VNI is the common binding factor for a L2 virtual network across the fabric.

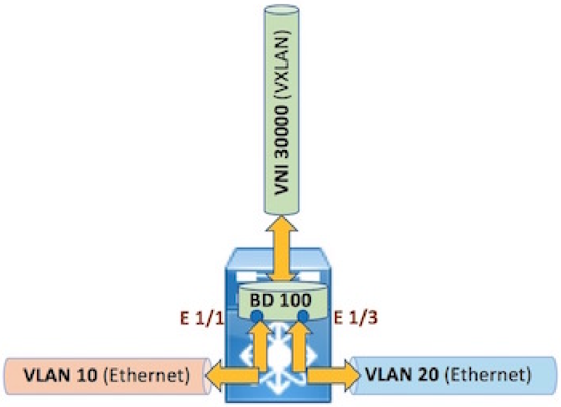
Note A bridge-domain or VLAN has no significance from a global identifier perspective when using port-VLAN translation.

Figure 60: Layer-2 traffic flow on a ToR switch



The colored lines represent bridged traffic in a Layer-2 virtual network (blue represents VNI 30000, green VNI 30001 and red VNI 30002), within the ToR switch. The dotted lines represent bridged traffic from the ToR switch to another switch.

Figure 61: Layer-2 multi-tenancy configuration



- In the above image, L2 VNI 30000 is represented by BD 100. VLAN 10 on port Ethernet 1/1 and VLAN 20 on port Ethernet 1/3 are tied to BD 100.

Note Though different ToR switches can have different bridge domain IDs for the same Layer-2 virtual network, it is convenient if you assign the same bridge domain ID.

A sample configuration for the above image is given below:

Note **Important**—Configurations in this chapter/white paper are partial configurations in the overall VXLAN BGP EVPN fabric configuration, and should not be done in isolation. For complete configuration, refer the *Forwarding Configurations* chapter, Cisco Nexus 5600 Series switch configuration and Cisco Nexus 7000 Series switch configuration sections.

Create the Layer-2 VNI on the ToR switch

(config) #

```
vni 30000
```

Create a bridge domain to represent the Layer-2 virtual network, and map it to the VNI

(config) #

```
system bridge-domain 100
bridge-domain 100
    member vni 30000
```

The **system bridge-domain** command identifies the bridge domain IDs and the **bridge-domain** command configures the specified bridge domain.

Create a profile that maps the 802.1Q VLANs to the Layer-2 VNI (through bridge domain 100)

(config) #

```
encapsulation profile vni vsi_10
   dot1q 10 vni 30000

 encapsulation profile vni vsi_20
   dot1q 20 vni 30000
```

On Nexus 7000 Series switches, a port is assigned as a trunk port by applying the above profile (mapping) on the specified port.

Assign the 802.1Q VLANs to a designated trunk port

(config) #

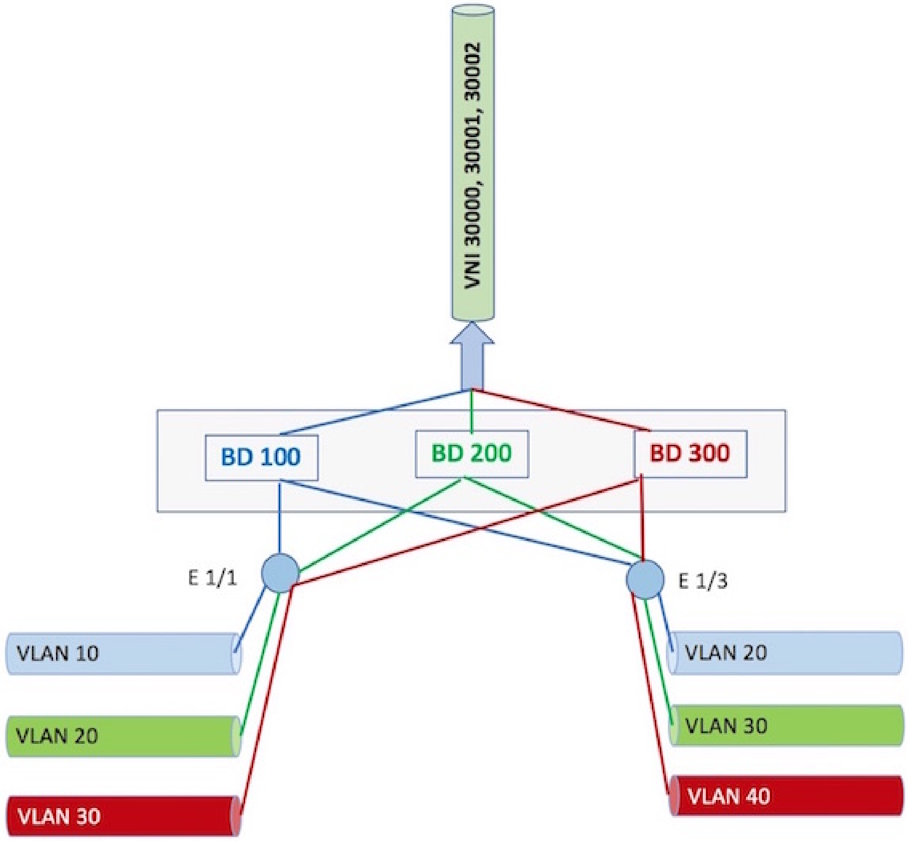
```
interface Ethernet 1/1
   no shutdown
   service instance 1 vni
       no shutdown
       encapsulation profile vsi_10 default

interface Ethernet 1/3
   no shutdown
   service instance 1 vni
       no shutdown
       encapsulation profile vsi_20 default
```

In VLAN mode, the **switchport mode trunk** command was used to designate a port as a trunk port.

Use case—Multiple Layer-2 Virtual Networks' configuration

Typically, multiple Layer-2 networks are represented on a ToR switch.

Figure 62: Multiple Layer-2 virtual networks on a ToR switch



You can see the flexibility of mapping from the above image.

- VLAN 10 on Ethernet 1/1, and VLAN 20 on Ethernet 1/3 are in the same BD/VNI (BD 100/VNI 30000).
- VLAN 20 on Ethernet 1/1, and VLAN 30 on Ethernet 1/3 are in the same BD/VNI (BD 200/VNI 30001).
- VLAN 30 on Ethernet 1/1, and VLAN 40 on Ethernet 1/3 are in the same BD/VNI (BD 300/VNI 30002).

A sample configuration for multiple Layer-2 networks' configuration at the same time is given below:

Create Layer-2 VNIs on the ToR switch

(config) #

```
vni 30000, 30001, 30002
```

Create bridge domains to represent the Layer-2 virtual networks, and map it to the respective VNIs

(config) #

```
system bridge-domain 100, 200, 300
bridge-domain 100, 200, 300
    member vni 30000 - 30002
```

In the above configuration, BD 100 is subscribed to VNI 30000, BD 200 to VNI 30001, and BD 300 to VNI 30002.

Create profiles that map 802.1Q VLANs to respective Layer-2 VNIs

(config) #

```
encapsulation profile vni vsi_10-30
   dot1q 10, 20, 30 vni 30000, 30001, 30002

encapsulation profile vni vsi_20-40
   dot1q 20, 30, 40 vni 30000, 30001, 30002
```

On Nexus 7000 Series switches, a port is assigned as a trunk port by applying the above profiles (and mappings) on the specified ports.

Assign the 802.1Q VLANs to designated trunk ports

(config) #

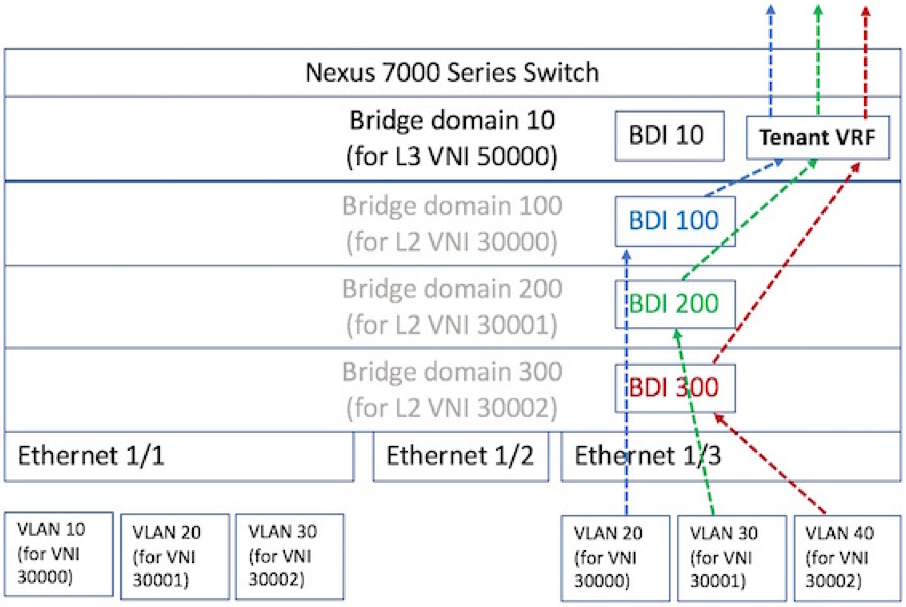
```
interface Ethernet 1/1
   no shutdown
   service instance 1 vni
       no shutdown
       encapsulation profile vsi_10-30 default

interface Ethernet 1/3
   no shutdown
   service instance 1 vni
       no shutdown
       encapsulation profile vsi_20-40 default
```

Layer 3 Multi-Tenancy on Cisco Nexus 7000 Series Switches

Earlier, the image *L2 traffic flow on a ToR switch* depicted Layer-2 traffic flow on a ToR switch. In the image *Layer-3 traffic flow* (below), the dotted colored lines represent routed traffic between 2 different Layer-2 virtual networks.

Note A bridge domain (BD) representation for Layer-3 VNI 50000 is added in the image.

Figure 63: Layer-3 traffic flow

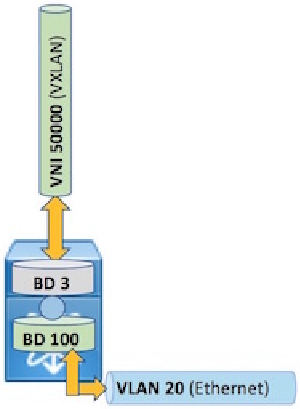


A virtual L3 interface, called a bridge domain interface (BDI), is used to route traffic from a BD. A BDI should be created for each BD (for each L2 VNI) and added to the tenant VRF.

Let us say the source end host in the Layer-2 virtual network 30000 is represented by VLAN 20, on port Eth 1/3. The source sends traffic to an end host on a different L2 virtual network, external to the switch. Since the BDI is added to the VRF table, traffic will be routed through the L3 network VNI that represents the tenant, to the target ToR switch,.

A sample set of configurations is given below:

Figure 64: Layer-3 multi-tenancy configuration



A sample configuration for the above image is given below:

Create the Layer-3 VNI for the tenant VRF

(config) #

```
vni 50000
```

Create a bridge domain to represent the Layer-3 VNI, and map it to the VNI

(config) #

```
system bridge-domain 3
bridge-domain 3
    member vni 50000
```

The **system bridge-domain** command identifies the bridge domain IDs and the **bridge-domain** command configures the specified bridge domain.

Configure a VRF overlay VLAN/BDI for the VRF

(config) #

```
interface BDI3
   no shutdown
   mtu 9216
   vrf member VRF-A
   ip forward
```

Create the tenant VRF and associate the Layer-3 VNI to it

(config) #

```
vrf context VRF-A
   vni 50000
   rd auto
   address-family ipv4 unicast
      route-target both auto evpn
```

Note In the Layer-2 multi-tenancy example, BD 100 was created to represent L2 VNI 30000 on the ToR/leaf switch, as shown below.

(config) #

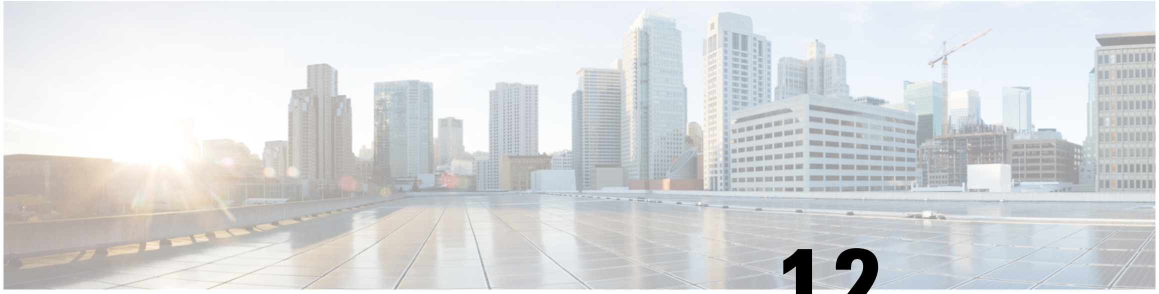
```
bridge-domain 100
    member vni 30000
```

Create a server facing BDI for BD 100 and associate it with the tenant VRF

(config) #

```
interface BDI100
  no shutdown
  vrf member VRF-A
  ip address 10.2.2.1/24
  fabric forwarding mode anycast-gateway
```

The virtual interface BDI100 is the Layer-3 interface that represents BD 100. This configuration updates the BDI information to the VRF table.



CHAPTER 12

Deploying Cisco Programmable Fabric

Deploying Cisco Programmable Fabric

This chapter briefly describes about the Cisco Programmable Fabric deployment and its requirements.

Platform Requirements

Cisco Programmable Fabric supports an IP fabric with VXLAN being employed as a network overlay and MP-BGP based EVPN address family being used for distribution of host information. MP-BGP-based EVPN address family distributes end host and external network reachability information in the fabric.

Following table shows the minimum hardware and software support required to deploy Cisco Programmable Fabric.

Table 6: Hardware and Software Support for Cisco Programmable Fabric with VXLAN BGP EVPN

Function	Platform	I/O Module	Minimum Software Release
Fabric Management [1]	Cisco DCNM	-	Cisco NX-OS 7.2(3)
Cisco Programmable Fabric Spine	Cisco Nexus 5600	-	Cisco NX-OS 7.3(0)N1(1)
	Cisco Nexus 7000/7700	F3 card	Cisco NX-OS 7.3(0)D1(1)
	Cisco Nexus 9200	-	Cisco NX-OS 7.0(3)I4(1)
	Cisco Nexus 9300	-	Cisco NX-OS 7.0(3)I1(3)
	Cisco Nexus 9500	-	Cisco NX-OS 7.0(3)I1(3)

Function	Platform	I/O Module	Minimum Software Release
Cisco Programmable Fabric Leaf	Cisco Nexus 5600	-	Cisco NX-OS 7.3(0)N1(1)
	Cisco Nexus 7000/7700	F3 card	Cisco NX-OS 7.3(0)D1(1)
	Cisco Nexus 9200	-	Cisco NX-OS 7.0(3)I4(1)
	Cisco Nexus 9300	-	Cisco NX-OS 7.0(3)I1(3)
	Cisco Nexus 9500	-	Cisco NX-OS 7.0(3)I1(3)
Cisco Programmable Fabric Border Leaf/Border Spine	Cisco Nexus 5600	-	Cisco NX-OS 7.3(0)N1(1)
	Cisco Nexus 7000/7700	F3 card [2]	Cisco NX-OS 7.3(0)D1(1)
	Cisco Nexus 9200	-	Cisco NX-OS 7.0(3)I4(1)
	Cisco Nexus 9300	-	Cisco NX-OS 7.0(3)I1(3)
	Cisco Nexus 9500	-	Cisco NX-OS 7.0(3)I1(3)

[1] We recommend you to carefully read the 'New and Changed Features' section on the Release Notes for additional information.

[2] Border PE is supported on Cisco Nexus 7000 Series Switches and Cisco Nexus 7700 Switches on (F3 line card) as a single box solution that does VXLAN decapsulation and MPLS encapsulation on the same box/VDC.

Note

1. Only Cisco Nexus 5600 Switches are supported for VXLAN. Cisco Nexus 5500 Switches and Cisco Nexus 6000 Series Switches are not supported.

2. For Cisco Programmable Fabric, we recommend at least one multiprotocol BGP route-reflector (RR). As an integrated function of Cisco Programmable Fabric, the following platforms can support this function:

    - Cisco Nexus 5600 Switches with Cisco NX-OS 7.3(0)N1(1) and later releases.

    -

Guidelines and Limitations for Cisco Programmable Fabric

Cisco Programmable Fabric has the following guidelines and limitations:

Guidelines

- To ensure that the day-zero POAP device provisioning does not interfere with other DHCP servers on your network, we recommend that you use a dedicated VLAN and subnet for the fabric management network. Cisco DCNM and the Ethernet out-of-band/inband ports of the Cisco Programmable Fabric switches (mgmt0 or Ethernet front panel ports) reside in the fabric management network. You have the option to interconnect the fabric management network with your existing out-of-band management network.

- Beginning with Cisco NX-OS Release 7.0(3)I5(2), inband POAP and management are available for Cisco Nexus 9000 Series switches and Cisco NX-OS 7.1(2) on Cisco DCNM with 10.1(2)ST(1) Cisco DCNM templates.
- Cisco Programmable Fabric supports POAP via the out-of-band and inband network, which means POAP via the management interface (mgmt0) and front panel Ethernet ports.
- For Cisco Programmable Fabric, auto-configuration feature is the reachability to the LDAP/network database through inband and as well through out-of-band support.
- Every Cisco Programmable Fabric switch that is managed by Cisco DCNM via out-of-band or inband network must be connected to the fabric management network through the Ethernet out-of-band or inband network.
- A console connection for fabric management is recommended but not required for Cisco Programmable Fabric.
- If Cisco DCNM is your repository server for POAP, you must upload the Cisco NX-OS kick start and system images to Cisco DCNM using the Secure Copy Protocol (SCP) or Secure File Transfer Protocol (SFTP).

Limitations

- The fabric management network can support only one Dynamic Host Configuration Protocol (DHCP) server. You can either use the DHCP server in Cisco DCNM or a designated DHCP server, but not both.
- You can create a VRF instance using a controller (Cisco APIC or Cisco DCNM) or using the CLI. If you want to delete a VRF instance that was created using a controller, you have to use the controller for deletion too, and not the CLI. However, for a controller created VRF instance, you can add VRF attributes such as route-map, etc, through the CLI.

How to Cable the Network Fabric and Servers for Cisco Programmable Fabric

Fabric Management Network and Console

We recommend that every Cisco Programmable Fabric switch that is to be managed by Cisco Programmable Fabric is directly connected (Layer-2) to the fabric management network through the Ethernet out-of-band port (mgmt0). Else you need to add specific configuration on the router (helper-address) and add a static route on the Cisco DCNM to reach the out-of-band network.

The Cisco DCNM is the central point of management for Cisco Programmable Fabric. For more information on Cisco DCNM installation, see Cisco DCNM Installation Guide.

Figure 65: Cabling the Fabric Management Out-of-band Network

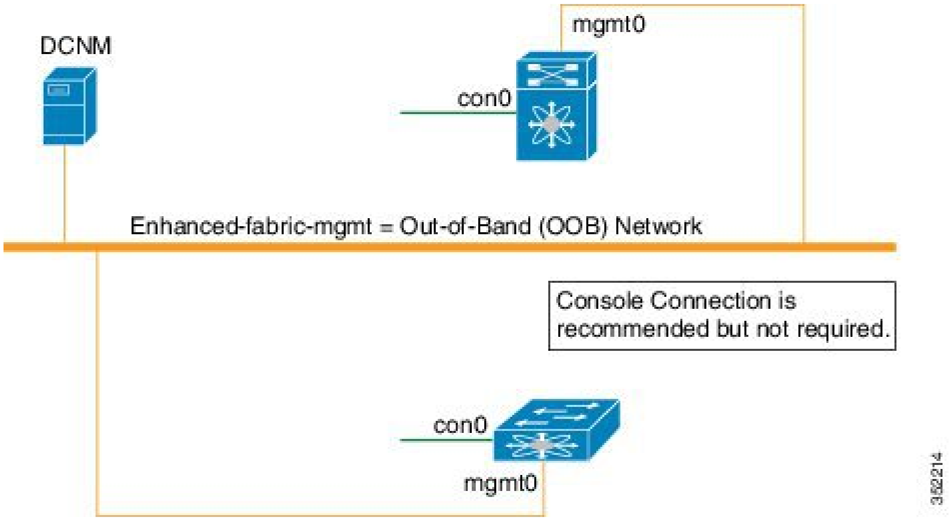
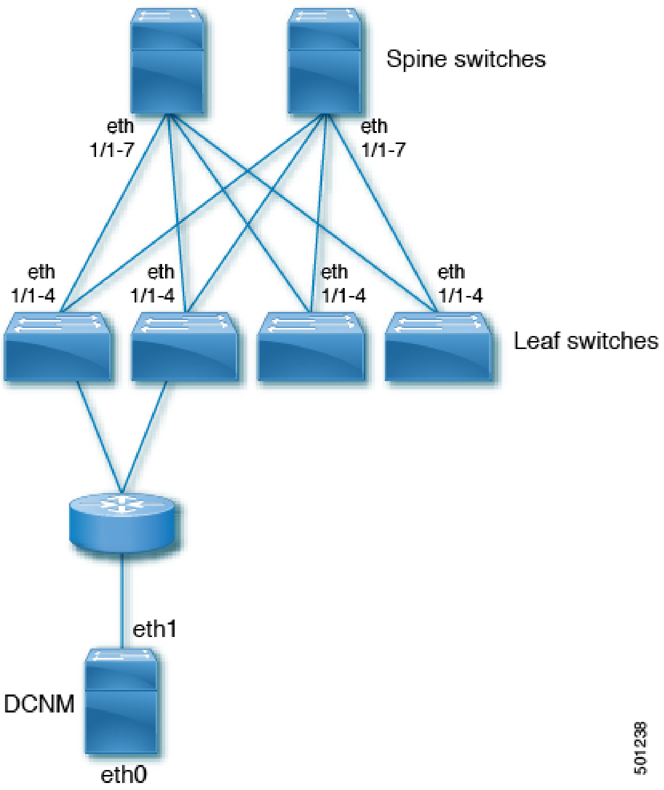

DCNM
mgmt0
con0
Enhanced-fabric-mgmt = Out-of-Band (OOB) Network
Console Connection is recommended but not required.
con0
mgmt0
352214

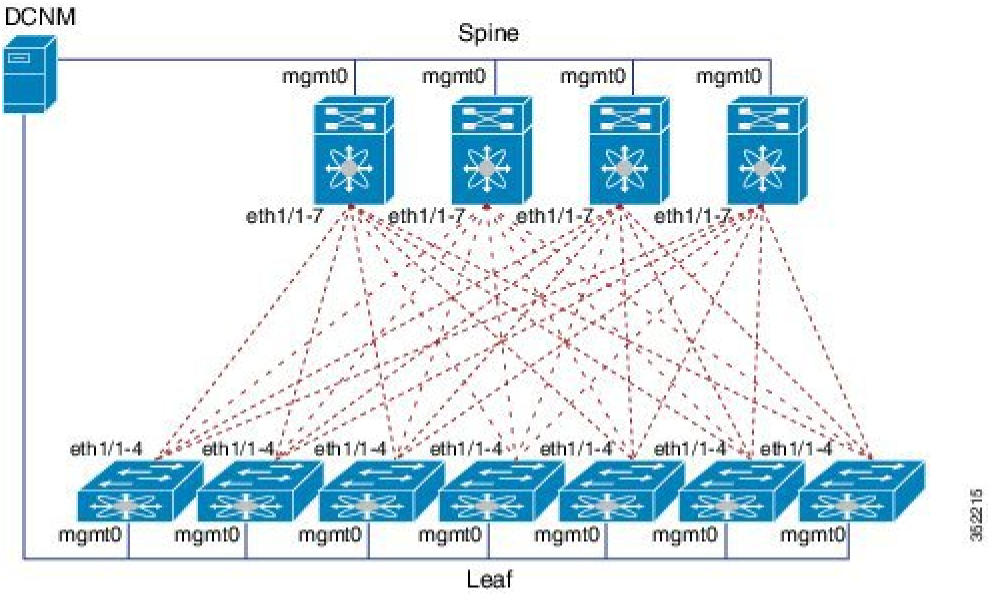
Figure 66: Cabling the Fabric Management Inband Network

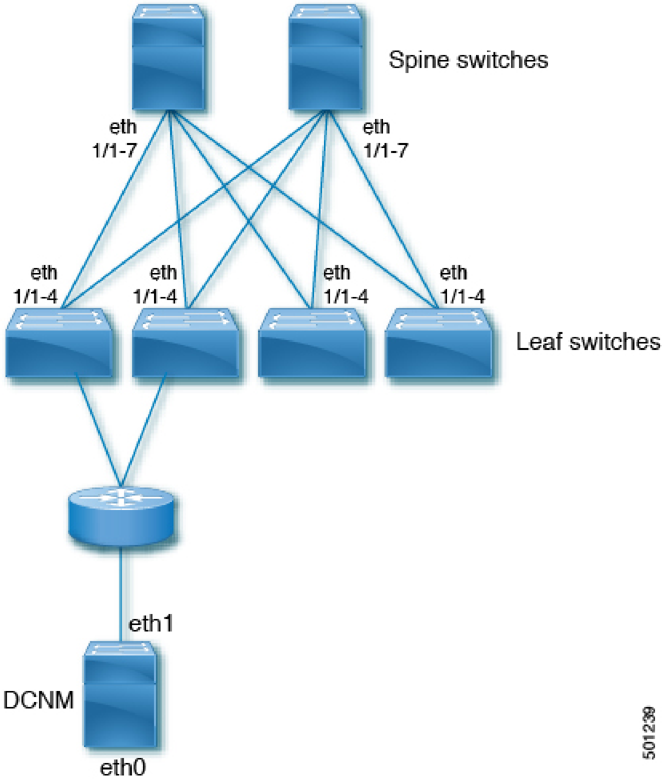


Fabric Connectivity

The fabric interfaces of the Cisco Programmable Fabric switches connect to one another. Fabric interfaces are configured as regular Layer-3 routed ports for efficient forwarding of traffic on the IP underlay. Choice of routing protocols includes Layer-3 IS-IS and Open Shortest Path First (OSPF). Both IP numbered and IP unnumbered options are supported.

Figure 67: Cabling the Cisco Programmable Fabric Networks and Servers



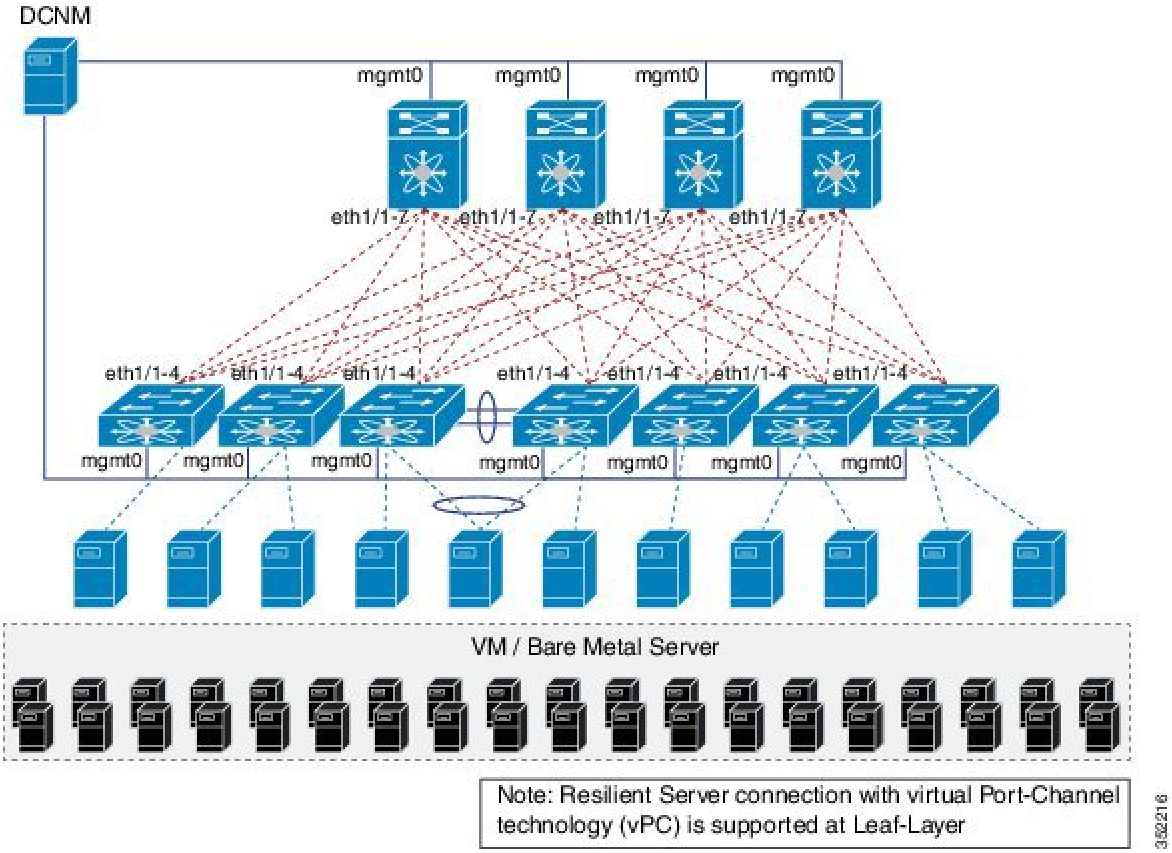


Server Connectivity

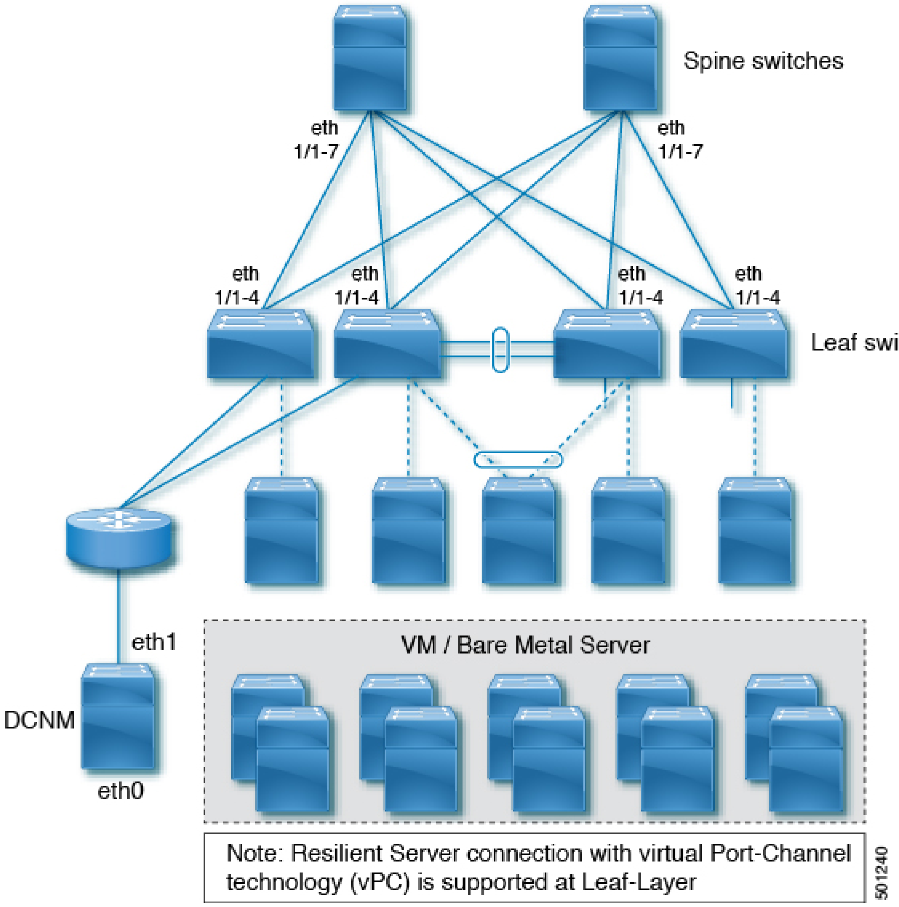
To transport data traffic across the Cisco Programmable Fabric, the leaf switch must receive the traffic for connected VLANs that are to be extended across the fabric. The leaf-to-server interfaces are called host interfaces.

Note Always connect servers to Cisco Programmable Fabric leaf switches. You must not connect servers to Cisco Programmable Fabric border leaf switches (since host-based auto-configuration is not supported) and spine switches.

Figure 68: How to Cable Server Connectivity



Note: Resilient Server connection with virtual Port-Channel technology (vPC) is supported at Leaf-Layer



Deploying Cisco Programmable Fabric

1. Ensure that you have the appropriate Cisco Nexus devices with the minimum required Cisco NX-OS software releases to support Cisco Programmable Fabric. For more information on hardware and software support, see Platform Requirements.

2. Install the Data Center Nexus devices. For more information on installation and upgrade, see the following appropriate guides for your Cisco Programmable Fabric switches.

   - Cisco Nexus 9000 Series Switches
   - Cisco Nexus 7000 Series Switches and Cisco Nexus 7700 Switches
   - Cisco Nexus 5600 Series Switches

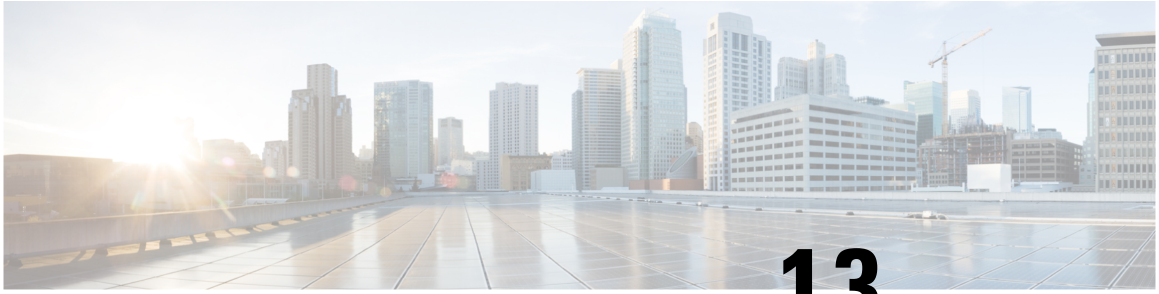
3. (Optional) Cisco Nexus 1000V is necessary, only if your requirement is to support VDP based auto-configuration. Install and configure the Cisco Nexus 1000V switch for VMware vSphere for Cisco Programmable Fabric. For more information on Cisco Nexus installation, see the Cisco Nexus 1000V Installation and Upgrade Guide.

Note To deploy Cisco DCNM, two port groups or port profiles are required on the virtual switch.

4. Install either Cisco DCNM Open Virtual Appliance (OVA) or Cisco DCNM Virtual Appliance (ISO). For more information on Cisco DCNM installation, see the Cisco DCNM Installation Guide.
    a. Install Cisco DCNM OVA to manage all the applications for the central point of management or
    b. Install Cisco DCNM ISO to deploy on ESXi and KVM Hypervisors
5. (Optional) Use the following option to install OpenStack for Cisco Programmable Fabric:
    - Install the Cisco OpenStack Installer to install the OpenStack for Cisco Programmable Fabric orchestrator.

Note

- Before installing the Cisco OpenStack installer, ensure to install Cisco Programmable Fabric, switches, Cisco DCNM OVA or Cisco DCNM ISO must be already installed.
- To support OpenStack for Cisco Programmable Fabric, Cisco DCNM must be accessible via the OpenStack controller and Cisco Programmable Fabric.



CHAPTER 13

Introducing Cisco Programmable Fabric Management, Operations, and Provisioning

Introduction to Programmable Fabric Management, Operations, and Provisioning

This chapter briefly describes POAP auto configuration profiles:

Power On Auto Provisioning in Programmable Fabric

A NX-OS device with Power On Auto Provisioning (POAP) feature enabled needs to be able to find its IP address and download image/configuration and successfully complete the POAP process via the DHCP server and a repository server or Cisco DCNM that is located across in a VXLAN/EVPN fabric.

In a VXLAN-EVPN based Programmable Fabric deployment, day-0 bring up of switches is supported in an automated way via POAP. Along with the traditional POAP option via the mgmt0 (out-of-band) interfaces, day-0 bring-up of the devices can also be done over the front-panel Ethernet (inband) interfaces.

Note Inband POAP and management are available beginning with Cisco NX-OS Release 7.0(3)I5(2) for Cisco Nexus 9000 Series switches and Cisco NX-OS 7.1(2) on Cisco DCNM with 10.1(2)ST(1) Cisco DCNM templates.

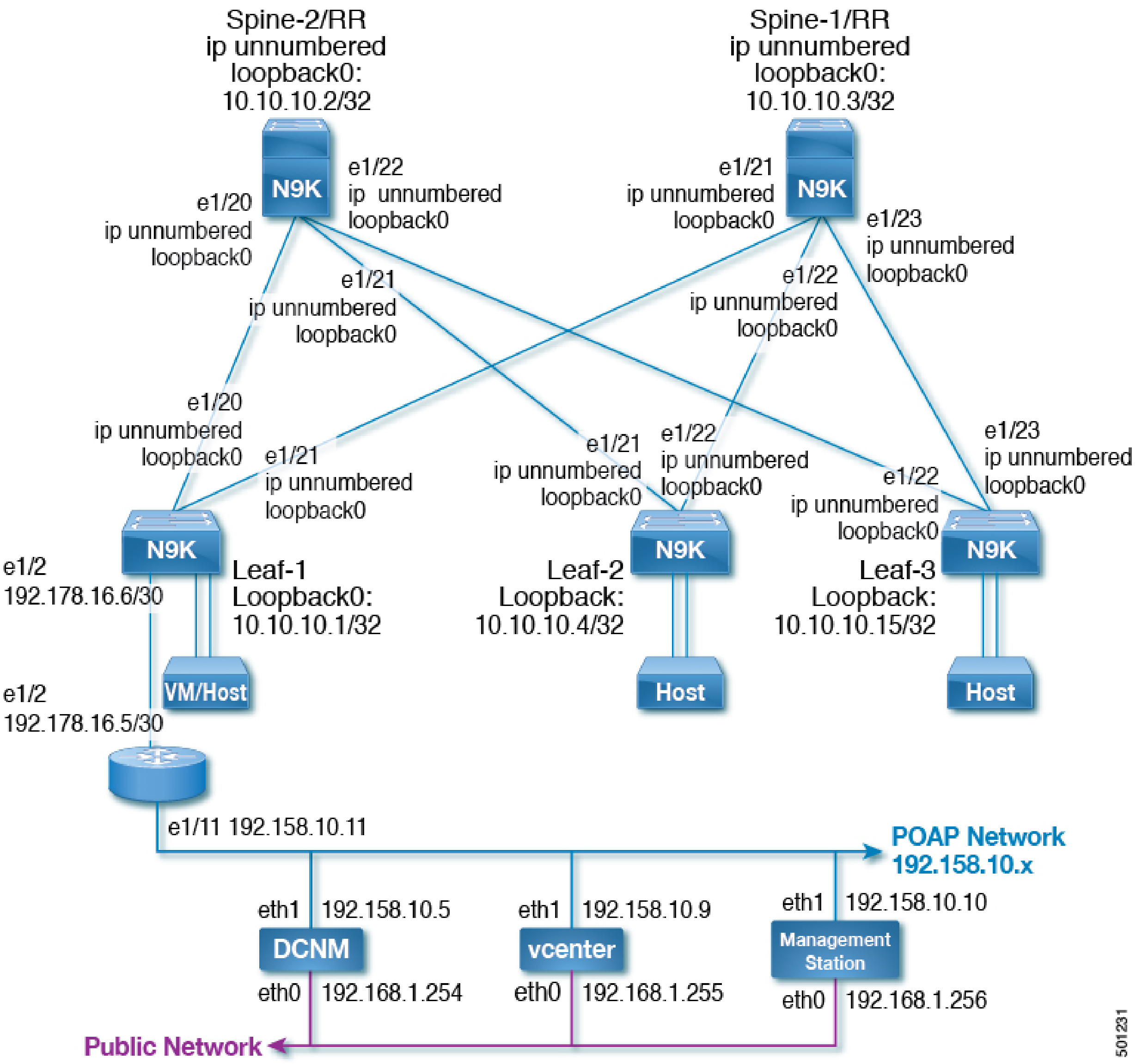
In an IP fabric, the use of IP unnumbered interfaces is preferred to simplify IP address management on the underlay where only one unique IP address per device is required to bring up the routing table state between the various devices. All the core facing interfaces on a device that participate in the IGP (for example, IS-IS or OSPF) share this per device unique IP. While it greatly simplifies the IP address management on the underlay, it certainly complicates how DHCP relay functionality works on the inband interfaces as there are no longer regular Layer 3 interfaces under which a relay can be readily configured. Without DHCP relay functionality, inband POAP cannot work.

POAP over IP numbered interfaces works when the DHCP server is configured with unique subnet scopes for every Layer 3 core-facing interface pair and IP DHCP relay is enabled under every core-facing interface. The DHCP server can be connected to a leaf that is attached to the default VRF.

Prerequisites for Inband POAP over IP Unnumbered Links

See the following illustration for inband POAP via IP unnumbered fabric interfaces:

Figure 69: Inband POAP via IP Unnumbered Fabric Interfaces



See the following prerequisites for inband POAP over IP unnumbered links:

- The seed/edge leaf **leaf-1** is the leaf that is connected to the POAP network. In the topology, a router sits in between the edge leaf and the DCNM.

- On the seed/edge leaf node, the IP address of the port connecting to the DCNM network is recommended to be a /30 IP. It means that the IP on the other side, the next hop, is the other IP in the /30 network.
- On the router connecting the seed leaf to the DHCP network, DHCP relay configuration for reachability to the DHCP server IP must be made. An example CLI configuration on a Cisco Nexus switch that acts as a router is **ip dhcp relay address** <*dhcp_server_ip*>.
- The DHCP server IP is the IP address of the DCNM if DCNM is used for POAP. Otherwise it is the IP address of the standalone DHCP server used. If DCNM is used for fabric deployment, the DCNM POAP template has the option to configure two DHCP Servers.
- For inband POAP, the DHCP server's **dhcpd.conf** must be configured manually to include new scope(s) for the networks in the fabric.
- On the DCNM, vCenter, or on any other Network Management Station in the internal network, the route configuration must be done manually for network reachability (192.178.16.0 and 10.10.10.0 networks in the above sample topology).
- There needs to be at least one seed/edge leaf node in the fabric with connectivity to the DCNM/DHCP server/vCenter/Management Station etc. for the other NX-OS nodes in the IP Fabric to successfully go through inband POAP.

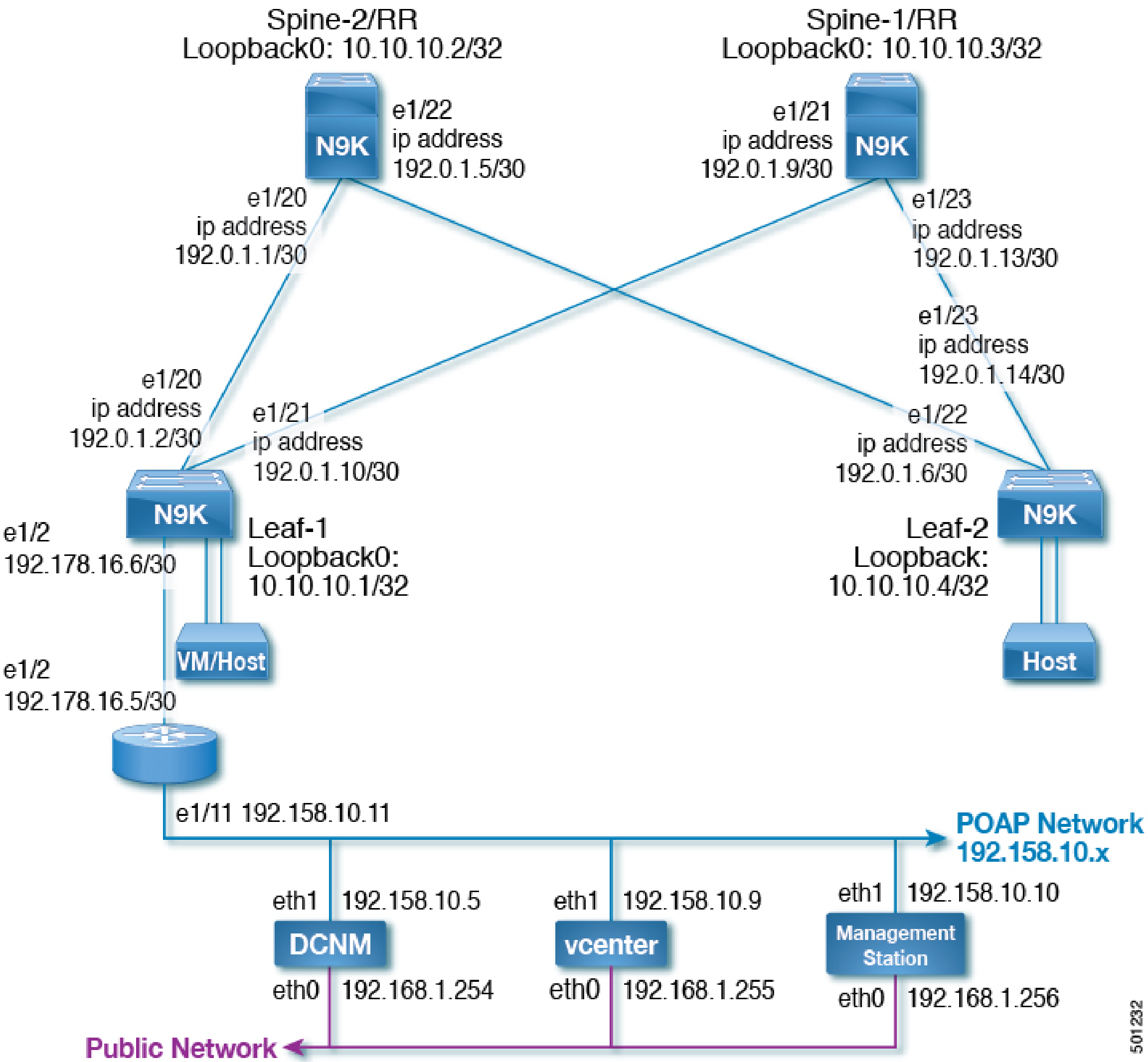
With DCNM as the DHCP and Repository/Configuration Server:

- When DCNM is used as the configuration server for the fabric, DCNM POAP templates are used for day-0 configuration on the devices. The DCNM POAP templates support the topology displayed in the illustration.
- For the devices to be discovered in the inband IP fabric topology on the DCNM, the **Management IP** field in the POAP template's **General** tab should be the management IP of the device which is its loopback0 IP. This is not the **mgmt0** interface IP as the vrf is **default** and not **management**.
- The first time the devices are being brought up, **write erase** and **reload** must be issued on the device and not via the DCNM GUI. It is because the management IP on DCNM for inband case is that of the loopback0 IP and the fabric is not discovered yet before POAP for reachability from DCNM.
- DHCP scope for the underlay network must be manually added either by configuring **DHCP Scope** via DCNM GUI or by editing the **dhcpd.conf** file in the DHCP server. Routes for reachability must also be manually added on other devices like the router in between, DCNM, and vCenter etc.
- The seed leaf needs to be brought up first, following which the RR needs to be brought up. After the seedleaf and RR spine are up, any of the other nodes (non-RR spine, other Leaf nodes) are ready to successfully finish inband POAP. The devices loop in POAP phase until the seed leaf, RR are up and until there is a successful DHCP handshake and a successful TFTP operation.

Prerequisites for Inband POAP over IP Numbered Links

See the following illustration for inband POAP via IP numbered fabric interfaces:

Figure 70: Inband POAP via IP Numbered Fabric Interfaces



See the following prerequisites for inband POAP over IP numbered links:

- The seed/edge leaf **leaf-1** is the leaf that is connected to the POAP network. In the topology, a router sits in between the edge leaf and the DCNM.
- On the seed/edge leaf node, the IP address of the port connecting to the DCNM network is recommended to be a /30 IP. It means that the IP on the other side, the next hop, is the other IP in the /30 network.
- On the router connecting the seed leaf to the DHCP network, DHCP relay configuration for reachability to the DHCP server IP must be made. An example CLI configuration on a Cisco Nexus switch that acts as a router is **ip dhcp relay address** <*dhcp_server_ip*>.
- The DHCP server IP is the IP address of the DCNM if DCNM is used for POAP. Otherwise it is the IP address of the standalone DHCP server used. If DCNM is used for fabric deployment, the DCNM POAP template has the option to configure two DHCP Servers.

- For inband POAP, the DHCP server's **dhcpd.conf** must be configured manually to include new scope(s) for the networks in the fabric. In the **IP Numbered** topology, the DHCP scope is required for every fabric link.

  Alternately, the following script is available as part of the DCNM OVA installation and it can be used to generate the DHCP scopes automatically on DCNM: **/root/utils/inband_p2p_dhcp_scope.py**. The script can be modified or customized if needed. The instructions are provided inside the script itself.

- On the DCNM, vCenter, or on any other Network Management Station in the internal network, the route configuration must be done manually for network reachability (192.178.16.0 and 192.0.1.0 networks in the above sample topology).
- There needs to be at least one seed/edge leaf node in the fabric with connectivity to the DCNM/DHCP server/vCenter/Management Station etc. for the other NX-OS nodes in the IP Fabric to successfully go through inband POAP.

With DCNM as the DHCP and Repository/Configuration Server:

- When DCNM is used as the configuration server for the fabric, DCNM POAP templates are used for day-0 configuration on the devices. The DCNM POAP templates support the topology displayed in the illustration.
- For the devices to be discovered in the inband IP fabric topology on the DCNM, the **Management IP** field in the POAP template's **General** tab should be the management IP of the device which is its loopback0 IP. This is not the **mgmt0** interface IP as the vrf is **default** and not **management**.
- The first time the devices are being brought up, **write erase** and **reload** must be issued on the device and not via the DCNM GUI. It is because the management IP on DCNM for inband case is that of the loopback0 IP and the fabric is not discovered yet before POAP for reachability from DCNM.
- DHCP scope for the underlay network must be manually added either by configuring **DHCP Scope** via DCNM GUI or by editing the **dhcpd.conf** file in the DHCP server. Routes for reachability must also be manually added on other devices like the router in between, DCNM, and vCenter etc.
- The seed leaf needs to be brought up first, following which the RR needs to be brought up. After the seedleaf and RR spine are up, any of the other nodes (non-RR spine, other Leaf nodes) are ready to successfully finish inband POAP. The devices loop in POAP phase until the seed leaf, RR are up and until there is a successful DHCP handshake and a successful TFTP operation.

Inband Management

Device management is done via **vrf management** in the out-of-band network or via **vrf default** in the inband network. This feature adds support to manage the network devices from DCNM via the front panel inband ports under **vrf default** for Cisco Nexus 9000 Series switches.

Note Inband POAP and inband management is available beginning with Cisco NX-OS Release 7.0(3)I5(2) for Cisco Nexus 9000 Series switches and Cisco NX-OS 7.1(2) on Cisco DCNM with 10.1(2)ST(1) Cisco DCNM templates.

Inband POAP with inband management (Inband POAP and vrf = **default** for management) and Out-of-band POAP with Out-of-band management (POAP over mgmt0 and vrf = **management** for management) are currently supported via the DCNM POAP templates.

DCNM POAP Templates

See the following templates that are created to help generate the required configuration for VXLAN EVPN feature for Cisco Nexus switches:

- Leaf
- Spine
- Border Leaf
- Border Spine

Note POAP is day-0 device automated provisioning. For information on the installation of DCNM POAP templates, see the Cisco DCNM Installation Guide. For POAP launchpad, see the Cisco DCNM Web Client Online Help.

POAP template parameters are divided into various groups and these parameters are used by the template to generate a set of CLI commands to be run on a switch.

Note The parameters may vary based on your POAP template selection.

The VXLAN EVPN leaf template supports Dynamic Virtual Ports (DVP) and multi-mobility domain with DCNM version 10.0(1) or later, but the DVP remains disabled by default.

You can configure the encapsulation settings from **Admin** > **Fabric** > **Fabric Encapsulation Settings** to specify the following settings:

- Enable VXLAN Encapsulation for Leaf Network
    - Specify multicast group subnet (default is 239.1.1.0/25)
    - Specify number of Rendezvous Points (RPs): 1, 2, or 4
        - One or multiple multicast groups based on the number of RPs specified are generated by Cisco DCNM automatically.
    - Specify RP Subnet (default is 10.254.254.0/24)
        - Phantom RP addresses are generated from the RP subnet
        - RP Redundancy is same as number of RPs
        - Corresponding loopback interfaces with IP addresses in the same subnet as the phantom RP addresses but with different masks are added to each RP.

Fabric Settings

The following global settings can be configured from the Cisco DCNM application.

You can specify the general settings for the Fabric. From the menu bar, select **Admin** > **Fabric** > **General Settings** to specify the following settings:

- Configure LDAP Server and Segment/Partition ID ranges
  - Segment ID Range
  - Partition ID Range
- Configure dynamic VLAN ranges used by auto configuration
  - System Dynamic VLANs
  - Core Dynamic VLANs
- Configure global anycast gateway MAC

You can set common parameters, that are populated as default values in POAP templates. For a new POAP template, values defined in this global settings page, are automatically pre-populated. From the menu bar, select **Admin** > **Fabric** > **POAP Settings** to specify the following settings:

- Configure global setting for the fabric backbone
  - BGP AS
  - BGP RR IP/IP
  - Backbone Prefix Length
  - Enable conversational learning

You can configure the encapsulation settings from **Admin** > **Fabric** > **Fabric Encapsulation Settings** to specify the following settings:

- Enable VXLAN encapsulation for leaf network
- Specify multicast group subnet (default is 239.1.1.0/25)
- Specify number of rendezvous points (RPs): 1, 2, or 4
- One or multiple multicast groups based on the number of RPs specified are generated by Cisco DCNM automatically
- Specify RP Subnet (default is 10.254.254.0/24)
- Phantom RP addresses are generated from the RP subnet
- RP Redundancy is same as number of RPs
- Corresponding loopback interfaces with IP addresses in the same subnet as the phantom RP addresses but with different masks are added to each RP

Note The inband POAP and management support is available on Cisco DCNM POAP templates for Cisco Nexus 9000 Series switches since Cisco DCNM 10.1(x) release.

You can configure the following fields in the templates for inband POAP template configuration on leaf and spine nodes:

Cisco Nexus 9000 Series switches Leaf POAP Template:

For Seed/Edge Leaf configuration:

- General->Management Vrf : Select **default** from the dropdown.
- General->Seed device to enable inband POAP and management.
- General->Inband Port
- General->Inband Local Port IP
- General->Inband Next-Hop Port IP
- General->DHCP Server IP Address
- General->Second DHCP Server IP Address

Spine POAP Template:

- General->Management VRF: Select **default** from the dropdown.
- Manageability->DHCP Server IP
- Manageability->Second DHCP Server IP Address

Auto-Configuration in Programmable Fabric

Configuration Profile

A configuration profile in Cisco Programmable Fabric is a collection of commands used to instantiate a specific configuration. Based on appropriate end-host triggers (VDP or data plane trigger (any data frame)), the configuration profiles are grouped to allow flexible and extensible options to instantiate day-1 tenant-related configurations on a Cisco Programmable Fabric leaf on need basis.

The commands are entered using variables for certain parameters instead of entering the actual value. The switch then populates the actual values to derive the completed command. When the required parameters for a particular configuration profile are available, the profile can be instantiated to create a configuration set. The switch then applies this configuration set to complete the command execution belonging to the configuration set.

The commands which are supported under a configuration profile are called config-profile-aware commands. Most of the commands in the switch can be entered into the configuration profile.

Note Various sets of configuration profiles can be created and stored in the network database, and each network can use a different configuration profile. The configuration profiles can be used from the network to set up on the leaf whenever required.

Profile Refresh

Profile refresh involves updating and/or removing profile parameters (arguments or variables) without disrupting the traffic while using universal profiles. After the changes are done, Cisco DCNM executes the **fabric database refresh vni/vrf** command.

LDAP Configuration

There are four different tables, which you can query from:

- Network Table
- Partition Table
- Profile Table
- Host Table
- BL-DCI Table

Network Table

All the parameters for a network host are stored in this table in the LDAP. ToR will query this LDAP table for network parameters based on the following configuration:

```
Switch# configure terminal
Switch(config)# fabric database type network
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=networks, dc=cisco, dc=com key-type 1
```

Partition Table

All the parameters that are required to provision a VRF on the ToR are stored in this table. When a network is using a Universal Profile, querying this table can identify corresponding VRF profile. ToR queries this table for every new network that is provisioned using Universal Profile. The following configuration helps ToR to query the partition table:

```
Switch# configure terminal
Switch(config)# fabric database type partition
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=partitions, dc=cisco, dc=com
```

Profile Table

Multi-tenancy lite version

This table is used to store configuration profiles, which are pre-packaged with DCNM and custom config profiles that are created. The Cisco Nexus 5000 Series Switches or Cisco Nexus 6000 Series Switches employ this tenancy model where a maximum of 4000 VLANs are supported on the ToR. So, if a profile is not pre-configured on the system then ToR will query profile table to download profile contents and cache it locally.

The following configuration helps ToR to query the profile table:

```
Switch# configure terminal
Switch(config)# fabric database type profile
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=profiles, dc=cisco, dc=com
```

Multi-tenancy full version

The Cisco Nexus 7000 Series Switches employ this tenancy model where, up to 4000 VLANs or dot1q tags can be supported on a per port basis.

The following configuration helps ToR to query the profile table:

```
Switch# configure terminal
Switch(config)# fabric database type profile
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=profilesBridgeDomain, dc=cisco, dc=com
```

Host Table

Host table contains the information about the host to network mappings. The following configuration helps ToR to query the host table:

```
Switch(config)# fabric database type host
Switch(config)# server protocol ldap host ip [ip address] vrf [vrf name]
Switch(config)# db-table ou=hosts,dc=cisco,dc=com
```

Note A host table is required for LLDP auto-config support.

BL-DCI Table

All the parameters for provision BL-DCI (See the DCI Auto-configuration section for details) are stored in this table in the LDAP. The following configuration helps the switches to query the BL-DCI table:

```
switch(config)# fabric database type bl-dci
switch(config)# server protocol ldap ip [ip address] vrf [vrf name]
switch(config)# db-table ou=bl-dcis,dc=cisco,dc=com
```

Note A BL-DCI table is required for border leaf. DCI Auto-configuration is not applicable to Cisco Nexus 9000 Series switches.

Specifying profile Mapping for network instances

It is necessary to specify how a particular network instance will be provisioned. That is either using network database lookup or by using a static Profile on the switch. The following configuration describes how to configure ToR to fetch network parameters from Remote database (LDAP).

```
Switch(config)# fabric database profile-map global
Switch(config-profile-map-global)# dot1q default dynamic
Switch(config-profile-map-global)# vni default dynamic
```

The following configuration is for DCI Profile.

```
Switch(config)# fabric database profile-map global
Switch(config-profile-map-global)# vrf default dynamic
```

LDAP Preemptive

Note LDAP Preemptive is currently not supported on Cisco Nexus 9000 Series switches.

Customer can deploy multiple LDAP servers. One server can be specified as primary server and the other servers become secondary. When the primary LDAP server is up, the device always queries the primary server. When the primary server is down, the device queries the secondary servers instead. However, when the primary server comes back, the device performs all new LDAP queries on the primary server again instead of sticking on the secondary. One of the reasons for this is for load-balancing, for example, half devices set LDAP-A as primary and half devices set LDAP-B as primary. Therefore, the load is evenly distributed between the two servers. Only if one server is down, all queries go to the server that is still up.

The following configuration specifies the primary LDAP server:

```
Switch(config)#fabric database server primary {ip <ipaddr> | ipv6 <ipv6addr> | host
<hostname>} [port <portnum>] [vrf {<vrf-name>}] [enable-ssl]
```

Note that the primary LDAP server specified in this CLI must have the same format as it is specified in the DB table configuration, for example, if the DB table configuration uses host name for the server, this CLI must use the host name. Similarly, if the DB table configuration uses the port/vrf/enable-ssl options, this CLI must use those options.

Routable Loopback Auto-Configuration

Note Routable loopback auto-configuration is not supported on Cisco Nexus 9000 Series switches.

In multi-tenant environments, every leaf switch needs to have a unique routable IP address per VRF for reachability. The user can use a knob to turn ON for an on demand for a per VRF basis. By turning on this knob, on every leaf switch where the VRF exists, there will be unique routable IP address allocated per leaf switch on that VRF (typically via a loopback interface). The reachability of this loopback address will be advertised via BGP-EVPN using route-type 5, thereby preventing any additional sources of floods or ARPs.

In many scenarios, there is a requirement of having a per leaf switch, per VRF, unique interface. Example scenarios that require this include DHCPv4 relay support when the DHCP server and client are in non-default VRFs behind a vPC pair of switches. In addition, this is a pre-requisite for a VXLAN OAM functionality. To automatically configure a loopback interface per leaf switch per VRF, the partition profile **vrf-common-evpn-loopback** can be used. During profile application, the leaf switches will automatically pick a free loopback interface ID, configure it with the respective VRF/partition, and associate the loopback interface with the IP address used for the VTEP associated loopback interface.

The following configuration results in the VRF profile being updated on the leaf switch resulting in the loopback routable IP address being auto-configured under that VRF as well as advertised via EVPN to all leaf switches.

```
configure profile vrf-common-evpn-loopback
interface loopback $system_auto_loopbackId
vrf member $vrfName
ip address $system_auto_backboneIpAddress/32 tag 12345
vrf context $vrfName

vni $include_vrfSegmentId rd auto
ip route 0.0.0.0/0 $include_serviceNodeIpAddress
 address-family ipv4 unicast
 route-target both auto
 route-target both auto evpn
 address-family ipv6 unicast
 route-target both auto
 route-target both auto evpn
router bgp $asn
vrf $vrfName
 address-family ipv4 unicast
  advertise l2vpn evpn
  redistribute direct route-map FABRIC-RMAP-REDIST-
  SUBNET maximum-paths ibgp 2
 address-family ipv6 unicast
  advertise l2vpn evpn
  redistribute direct route-map FABRIC-RMAP-REDIST-
  SUBNET maximum-paths ibgp 2
interface nve $nveId
   member vni $include_vrfSegmentId associate-vrf
!
```

DCI Auto-Configuration

Following are the DCI auto configuration supported for Cisco Programmable Fabric.

LISP

Locator/ID Separation Protocol (LISP) is routing architecture that provides new semantics for IP addressing. Following is the day zero configuration done on a switch.

```
feature lisp

vrf context <COMMON-RLOC-VRF>
 ip lisp itr-etr
 ip lisp itr map-resolver <MR-address>
 ip lisp etr map-server <MS-address> key <registration-key>
```

Following is the DCI auto configuration for a per partition configuration on a DC edge device.

```
configure profile lispDCEdgeNodeProfile
vrf context $vrfName
 ip lisp itr-etr
 ip lisp locator-vrf $COMMON-RLOC-VRF-NAME
 lisp instance-id $include_dciId
register-route-notifications tag $asn

 lisp dynamic-eid $vrfName
 database-mapping 0.0.0.0/0 $include_RLOC1-address, $include_RLOC1-priority,
$include_RLOC1-weight
 database-mapping 0.0.0.0/0 $include_RLOC2-address, $include_RLOC2-priority,
$include_RLOC2-weight
```

Layer-3 DCI for VXLAN EVPN

Layer-3 DCI supports both single box and two box solutions with EVPN to VRF Lite handoff at border leaf.

Following is a DCI one box configuration.

```
config profile vrf-common-universal-evpn-bl-dc-edge
vrf context $vrfName
  vni $include_l3SegmentId
  rd auto
  address-family ipv4 unicast
    route-target both $rsvdGlobalAsn:$dciId
    route-target both auto evpn
  address-family ipv6 unicast
    route-target both $rsvdGlobalAsn:$dciId
    route-target both auto evpn

router bgp $asn
vrf $vrfName
```

```
   address-family ipv4 unicast
                allow vpn default-originate
   address-family ipv6 unicast
                allow vpn default-originate

interface nve $nveId
      member vni $include_l3SegmentId associate-vrf
end

configure profile defaultCombinedEvpnBorderLeafDcEdgeProfile
  router bgp $asn
  include profile any
```

Following is a DCI two box configuration.

```
config profile vrf-common-universal-evpn-bl
vrf context $vrfName
  vni $include_l3SegmentId
  rd auto
  address-family ipv4 unicast
    route-target both auto evpn
  address-family ipv6 unicast
    route-target both auto evpn

router bgp $asn
   vrf $vrfName
       address-family ipv4 unicast
          maximum-paths 2
          maximum-paths ibgp 2
       allow vpn default-originate

     address-family ipv6 unicast
       maximum-paths 2
       maximum-paths ibgp 2
       allow vpn default-originate

   neighbor $include_peerIpAddress1 remote-as $include_peerAsn
       address-family ipv4 unicast
         send-community both

     neighbor $include_peerIpv6Address1 remote-as $include_peerAsn
       address-family ipv6 unicast
         send-community both

     neighbor $include_peerIpAddress2 remote-as $include_peerAsn
              address-family ipv4 unicast
         send-community both

     neighbor $include_peerIpv6Address2 remote-as $include_peerAsn
       address-family ipv6 unicast
         send-community both
end
```

Auto-Configuration Triggers

The Cisco Programmable Fabric supports the following triggers for VLAN, VRF, and SVI instantiation. The leaf switch will query LDAP database for configuration information specific to this VRF and/or VLAN/segment. Based on the response received the leaf switch will auto configure server facing SVI. If this is the first SVI for the VRF it belongs to, VRF configuration is also applied.

Following triggers are supported:

CLI (Static-host) Trigger

Beginning with Cisco NX-OS Release 7.0(3)I7(1), CLI dot1q based auto-configuration (auto-pull), CLI VNI with overwrite-vlan based auto-configuration (auto-pull), and CLI VNI based auto-configuration (auto-pull) is not supported. Instead, a new configuration command is introduced to support the persistence for the CLI triggered auto-configuration. Now you can use static host VNI configuration with an optional overwrite-vlan.

Use the following CLI configuration command to add the static host:

```
fabric database static-host
vni <vni-id>
    overwrite-vlan <ow-vlan-id>
    interface <interface-id>
```

Note If the overwrite-vlan option is used to specify the overwrite-vlan, the overwrite-vlan command should precede the interface command, for the overwrite-vlan to take effect.

Use the following CLI commands to add the static host configuration for a particular VNI:

```
fabric database static-host
  vni <vni-id>
    interface <interface-id>
```

For example, the following CLI command adds the static host for VNI 8000 at interface e1/2 and triggers a host add:

```
fabric database static-host
  vni 8000
    interface e1/2
```

You can specify multiple interface IDs under the same VNI, if they share the same VNI.

Use the following commands to remove the static host for a particular VNI:

```
fabric database static-host
  no vni <vni-id>
```

Note The CLI command **no fabric database static-host** is not allowed. Delete the individual entries to delete the static hosts.

Note that the above command removes all hosts under multiple interfaces if more than one interface is added under the same VNI. Alternatively, one can also remove a specific interface under the VNI (if multiple interfaces are under the same VNI) using the following command:

```
fabric database static-host
```

```
     vni <vni-id>
           no interface <interface-id>
```

You can check the current static host running configuration by executing the following command:

```
# show run fabric forwarding

!Command: show running-config fabric forwarding
!Time: Wed Jul  5 20:42:17 2017

version 7.0(3)I7(1)
feature fabric forwarding

fabric forwarding anycast-gateway-mac 0000.abcd.dcba
fabric database mobility-domain md0
fabric database profile-map global
  ethernet-tag encapsulation dot1q default dynamic
  ethernet-tag encapsulation vni default dynamic
  vdp vni default dynamic
  vdp dot1q default dynamic
  vrf default dynamic
fabric database static-host
  vni 11700
    interface Ethernet1/1
fabric database timer cleanup 5


interface Vlan1800
  fabric forwarding mode anycast-gateway
```

Use the **show fabric database static-host** command to display the status of the configured static hosts as displayed in the following example:

```
show fabric database static-host

                                            Retry
VLAN/VNI Interface       State      Delay (sec) Count
-------- --------------- ---------- ----------- --------
   11700 Ethernet2/3     Retry      10          [#00001]
   11701 Ethernet2/3     Active
```

Note After ISSU from an older release to Cisco NX-OS Release 7.0(3)I7.(1), older VDP and VMT host profiles are migrated seamlessly to the new software release. However, the older auto-pull exec commands have been deprecated in the new release and they are replaced by the new static-host command. The auto-pull profiles from the older release are recovered after an ISSU to Cisco NX-OS Release 7.0(3)I7(1). Note that you have to issue the **clear fabric database host** command to remove the older auto-pull entries after ISSU, replace them with equivalent static host configuration commands, and save the configuration.

Data Frame Snooping (DFS)/Dot1q Trigger

Auto-configuration is triggered when tagged tenant data traffic packets ingress the leaf switches. The packet VLAN is used to trigger the auto-configuration.

Note This functionality is not supported on Cisco Nexus 9000 Series switches.

The DFS auto-configuration is enabled on the switch by default. The following global configuration CLI can be used to control the state:

```
switch(config)# platform fabric database dot1q ?
  disable  Disable dot1q auto detection
  enable   Enable dot1q auto detection
```

LLDP Trigger

Note This functionality is currently not supported on Cisco Nexus 9000 Series switches.

Link Layer Discovery Protocol (LLDP) is used by network devices for advertising their identity, capabilities and neighbors on IEEE 802 local area network, principally wired Ethernet. LLDP information is sent by devices from each of their interfaces at fixed time interval in the form of Ethernet frame. The frame ends with special TLV, named end of LLDPDU. LLDP only interacts with HMM in triggering auto-configuration. Based on the response from HMM, LLDP manipulates VLAN on the interface.

In a multi-tenancy network, tenants can be dynamically added or removed from the network. Auto-configuration provides the ability to dynamically allocate or deallocate resources for every tenant. Auto-configuration helps with generating and executing the necessary configuration CLI for enabling a tenant to be connected to the network. Switch can support only a subset of tenants at a time, depending on the necessity of network connectivity.

For example, creating the VNI for the tenant, allocating the resources (such as a bridge-domain), hardware resources (such as PVTCAM), and so on.

Auto-Configuration for Bare-Metal Severs

Auto-Configuration for Bare-Metal Severs provides a touchless orchestration scheme with the repository hosted on Cisco DCNM that has the MAC to segmentId mapping. LLDP running either on the server or on the CNA attached to the server, the MAC address of the server will be notified to the leaf or ToR, which in turn should trigger auto-configuration for the same via this notification.

The following command helps to enable or disable the LLDP auto-configuration feature in the config mode:

```
[no] lldp fabric auto-config
```

The following show command displays all the ports for the triggered auto-configuration feature.

```
Switch(config)# show lldp fabric auto-config

Interface         PORT-CHANNEL  Mac-Address      VNI     VLAN    Port-Mode   Status
-------------------------------------------------------------------------------------
Ethernet1/30      Po100         8478.ac1b.70c4   40000   200     native      ADD SUCCESS
Ethernet1/29      NA            8478.ac1b.70c1   -       100     native      ADD SUCCESS
```

Virtual Machine Tracker Auto-Configuration

Virtual Machine Tracker auto-config is a feature that automatically configures a tenant for provisioning where the leaf switches listen to VMWare vCenter events and perform required auto-configuration.

Virtual Machines (VMs) on a single physical server can belong to different Layer-2 domains and can require the Cisco Nexus switches to provision multiple VLANs on the ports that connect to the physical servers. To provide seamless movement of VMs from one physical server to another, the servers must be reachable through the same Layer-2 domain so that the VMs can retain their IP addresses and network connectivity even after moving. A static predefined configuration requires provisioning of all possible VLANs that can be used by VMs in a server management domain on each port of the switch. This process can result in more logical port-based VLANs than the switch can support. Alternatively, you can dynamically provision VLANs on a switch, Layer-2 Ethernet or Layer-2 port-channel interface (attached to a server) based on the tracking of the VMs that are connected to the port and the VLAN requirements of these VMs.

Virtual Machine Tracker (VM Tracker) enables you to do the following:

- Identify the switch port that is used for each VM
- Identify the VLAN requirements of each VM
- Track the movement of VMs from one host (ESXi) to another
- Track VM configuration changes such as additions, deletions, or modifications of VLANs, and configure VLANs on switch ports accordingly
- Track whether the VMs are in the power on or power off state, and configure VLANs on local switch ports for a VM only if the VM is in the power on state

Note For all ports on which VM Tracker is enabled, you must not perform any Layer 2 or Layer 3 configuration that is related to switchports and VLANs.

Virtual Machine Tracker and VMware vCenter

VM Tracker connects with VMware vCenter and collects information about the VMs that are connected to each host. This information includes the number of VMs that are connected to the host and the switch port through which the VM receives network traffic.

After you enable VM Tracker on the switch, it automatically extracts the following information about VMs from vCenter:

- The host on which the VMs exist
- The switch ports through which the VM traffic flows
- The virtual network interface card (vNIC) that connects the VM to a virtual switch
- The power state of the VM
- The VLAN information of port groups or distributed virtual switch (DVS) port groups
- The port groups or DVS port groups that are required for the VM

Virtual Machine Tracker Auto-Configuration Workflow

The following is a typical workflow to provision a tenant and provide an EVPN capable fabric connection to the tenant's workloads.

Note All leaf switches in the network fabric are provisioned with the VM Tracker connections based on the set of hosts that are connected to the switch.

- Identify the VLAN that a tenant’s virtual machine needs to utilize.
- Provision the required port-groups on the VMware vSphere Distributed Virtual Switch (DVS) or virtual switch.
- Determine the configuration profile that needs to be associated with the tenant for provisioning the VRF. For example, associating the vrf-common-universal-evpn profile.
- Collect the tenant's network information and provision the network database with an application that uses the documented DCNM REST APIs. The network information includes:
    - Required configuration profile associated with the server-facing network that is being created.
    - All the network parameters required to provision the required configuration profile.
    - The mobility domain that is associated with the network. At this point, the fabric is ready for auto provisioning the workloads at any switch which is configured with the suitable mobility domain and the VM Tracker connection.

    Note Only one mobility domain can be supported.

- When the workloads are powered on, VM Tracker detects the workload and pulls the appropriate network information from the LDAP database and provisions the network on the switch where the workload is connected.
- When a workload is no longer necessary, powering the workload off ensures that the previously configured provisioning is removed.
- VM Tracker on Cisco Nexus 5000 and 6000 Series switches do not update the interface trunk allowed VLANs.

VM Tracker in VPC

You must configure VM Tracker on both the vPC switches.

In a multi-tenancy full VPC setup, VPC secondary will not trigger auto-configuration. The auto-configuration profile is synced from primary to standby.

If primary vPC interface is down, there will be no actions from primary vPC side. But when VM Tracker retry timer expires on secondary, the skipped profile is inspected and VM Tracker triggers auto-configuration from standby if it detects that the primary vPC interface is down. When both vPC pair interfaces are shut, the profile should be removed.

In a multi-tenancy full vPC setup, VM Tracker sends triggers on both sides. The host trigger is sent along with the vdc_mac to avoid race conditions in HMM where the same host sends triggers on both the peers.

Guidelines and Limitations for Virtual Machine Tracker Auto-Configuration

Virtual Machine Tracker has the following guidelines and limitations:

- The switch must be able to reach VMware vCenter over IPv4. See the Deploying Cisco Programmable Fabric section for details. IPv6 cannot be used for connectivity from switch to vCenter. This is a limitation for Cisco Nexus 9000 Series switches only.)
- Both hosts that are directly connected and connected through UCS-FI are supported. Only LLDP protocol is supported if the hosts are connected through UCS-FI and FEX cannot be in between the switch and FI.
- LLDP based VM Tracker is not supported on Cisco Nexus 7000 Series Switches and Cisco Nexus 7700 Switches (for Multi-tenancy full setup) for 7.3 release.
- Current version of VM Tracker:
    - Supports ESXi 5.1 and 5.5. For Cisco Nexus 9000 Series switches, ESXi 6.0 is also supported.
    - Supports only VMware orchestration, other orchestrations like SCVM is not supported
- CDP/LLDP enable/disable, interface shut/no shut:
    - When CDP/LLDP is disabled or interface shut, VM Tracker triggers profile unapply when sync-full-timer expires. When CDP/LLDP is enabled or interface no shut, VM Tracker triggers profile apply when find-new-host timer expires. Both sync-full-timer and find-new-host timer have default expiration interval of 3600 seconds.
- Enabling the VM Tracker auto-config feature is disruptive. A best practice is to disconnect all VM Tracker connections before enabling VM Tracker auto-config.
- When VM Tracker auto-config is enabled after VM Tracker is already connected to VMware vCenter and has configured the switch, the existing VM Tracker configuration is removed and then VM Tracker auto-config is triggered.
- When VM Tracker auto-config is disabled, the auto-config triggered configuration is removed and the VM Tracker reverts back to the configured VLAN that was initially created by VM Tracker.
- The VLAN is always created when the VM Tracker auto-config triggers the configuration. The auto-vlan enable command is ineffective and is not supported when VM Tracker auto-config is enabled.
- Allowed-vlan takes effect when VM Tracker auto-config is enabled.
- When migrating from VM Tracker to VM Tracker auto-config in a vPC setup, the following procedure is a best practice:
    - Disconnect all VM Tracker connections on the vPC primary and the vPC secondary switches.
    - Enable VM Tracker auto-config using the **vmtracker fabric auto-config** command on the vPC primary and the vPC secondary switches.
    - Connect the VM Tracker connection on the vPC primary and vPC secondary switches.

Enabling Virtual Machine Tracker Auto-Configuration

Step 1 Enters global configuration mode:

switch# **configure terminal**

Step 2 Enable VM Tracker feature.

switch(config)# **feature vmtracker**

Step 3 Enables VM Tracker auto-config trigger. The no form of the command disables the auto-config trigger:

switch(config)# **[no] vmtracker fabric auto-config**

Step 4 Enters VM Tracker connection configuration mode for the connection name specified. The no form of the command disables the connection:

switch(config)# **[no] vmtracker connection connection-name**

Step 5 Configures remote IP parameters:

switch(config-vmt-conn)# **[no] remote** {*ip address ip_address* | *port port_number* | *vrf*}

Step 6 Verifies the username and password to connect to vCenter:

switch(config-vmt-conn)# **username** *username* **password** *password*

Step 7 Connects to vCenter. The no form of the command disconnects VM Tracker from vCenter:

switch(config-vmt-conn)# **[no] connect**

Step 8 Displays the VM Tracker auto-configuration information:

switch(config-vmt-conn)# **show vmtracker fabric auto-config**

Example

The following example shows how to enable the VM Tracker auto-configuration trigger, followed by a verification command that displays the VM Tracker auto-configuration information:

```
configure terminal
 feature vmtracker
 vmtracker fabric auto-config
 vmtracker connection v229
  remote ip address  172.29.21.29 port 80 vrf management
  username user1 password abc1234
  connect


switch(config-vmt-conn)# show vmtracker fabric auto-config
Fabric Auto Configuration is enabled
Auto Configure Retry Time left: 107 seconds
Switch Device: SAL1833YM0V
------------------------------------------------------
Port         Port-Channel    Vlan    Status
------------------------------------------------------
```

```
Ethernet1/3  port-channel13   50     Pending
Ethernet1/3  port-channel13   56     Pending
```

VDP Trigger

Dynamic provisioning simplifies the management of the VRF and VLAN/BD configurations. Dynamic provisioning can be triggered by:

- VDP signaling from the server

Per-port configuration can be used to enable or disable dynamic provisioning. When the VM comes up in a leaf switch for the first time, the port receives an ARP or ND packet and for cases where VDP is enabled, the hypervisor may issue a VDP packet. The platform is expected to punt these packets to the HMM component, which will then install a tenant profile based on the information in this packet. In the simplest case, a (Port, VLAN) information from the incoming packet will be used to derive the tenant profile that needs to be installed. The following needs to be implemented when the Vinci functionality is enabled on the switch to support this feature:

- The server-facing port needs to be configured with 'default VSI' and 'auto-config' must be enabled. The Ingress CBL state must be set to allow packets in all the VLANs. Note that for DHCP based snooping, 'feature dhcp' should be enabled.
- Global ACLs must be set up to punt the packets of interest such as ARP and ND to the HMM component. All the remaining packets must be dropped to retain the CBL behavior. The global ACLs must be used only for the (Port, VLAN) that has not been configured on the system.

Once the tenant profile is installed, the VLAN/BD configuration along with the associated port membership will be configured.

The following is an example for VN-Segment configuration:

```
system bridge-domain 2-3967

feature vni
 vni 5000, 10010-10011, 20000

 system fabric bridge-domain 3001-3967
 bridge-domain 2,10-11

vrf context management

 encapsulation profile vni all_tenants
 dot1q 6-204 vni 7002-7200
 encapsulation profile vni cisco_all
 dot1q 10-1010 vni 10010-11010
 bridge-domain 2,10-11
 member vni 5000,10010-10011

interface Ethernet3/6
 no shutdown
 service instance 3 vni
 no shutdown
 encapsulation profile cisco_all default
```

The following is an example for VN-Segment dynamic auto-configuration:

```
interface ethernet 2/2
  service instance vni default
      encapsulation dynamic vdp


interface ethernet 2/2
  service instance vni default
      encapsulation dynamic frame-snoop profile cisco
```

Auto-configuration Coexistence

Network auto-configuration occurs from multiple triggers. Supported co-existence defines the set of compatible triggers. Co-existence requires both the VLAN and MD and VNI based LDAP entries with identical profile name and contents.

Per-Port Auto-Configuration Trigger

For auto-configuration to occur, interfaces connecting to the host or server workloads must be configured to specify the desired auto-configuration trigger.

Note This section is applicable only for Cisco Nexus 5600 Switches. Per-port auto-configuration is not applicable for Cisco Nexus 9000 Series switches.

Auto-configuration trigger is not enabled on the interface by default. The auto-configuration trigger must be explicitly configured on the interface and only one auto-configuration trigger can be configured per interface. Configuration is supported on CE Trunk (switchport mode trunk) interfaces only. VPC interfaces must be configured consistently. You must ensure the configuration consistency.

The following example shows how to enable the interface for dot1q, vdp, lldp or vmtracker. To change the trigger from one interface to another, ensure to disable the previous interface configuration.

```
[no] encapsulation dynamic {dot1q | vdp | lldp | vmtracker}
```

VXLAN OAM

VXLAN operations, administration, and maintenance (OAM) is a protocol for installing, monitoring, and troubleshooting Ethernet networks to enhance management in VXLAN based overlay networks.

Similar to utilities such as ping and traceroute or pathtrace that allow quick determination of problems in IP networks, equivalent troubleshooting tools have also been introduced to diagnose problems in VXLAN networks. VXLAN OAM tools like ping, pathtrace and traceroute provide the reachability information to VMs and VTEPs in a VXLAN network. There is a notion of OAM channel is used to identify type of VXLAN payload present in these OAM packets.

There are two types of payloads supported:

- Conventional ICMP packet to the destination to be tracked

- Special NVO3 OAM header that carries useful information

ICMP channel helps to reach the traditional hosts or switches that do not support these new OAM packet formats. Nvo3 channels helps to reach the supported hosts or switches and carries important diagnostic information as well. VXLAN NVO3 OAM messages may be identified via the reserved OAM ethertype or by using a well-known reserved source MAC address in the OAM packets depending on the implementation on different platforms. This constitutes a signature for recognition of VXLAN OAM packets. The VXLAN OAM tools are categorized as shown in table below.

Table 7: VXLAN OAM Tools

Category	Tools
Fault Verification	Loopback Message
Fault Isolation	Path Trace Message
Performance	Delay Measurement, Loss Measurement
Auxiliary	Address Binding Verification, IP End Station Locator, Error Notification, OAM Command Messages, and Diagnostic Payload Discovery for ECMP Coverage

Loopback (Ping) Message

Loopback message (Ping and Loopback messages are the same and used interchangeably in this document) is used for fault verification. The loopback message utility is used to detect various errors and path failures. Consider the topology in figure below where there are three core (spine) switches labeled Spine 1, Spine 2, and Spine 3 and five leaf switches connected in a Clos topology. The path of an example loopback message initiated from Leaf 1 for Leaf 5 is shown when it traverses via Spine 3. When the loopback message initiated by Leaf 1 reaches Spine 3, it forwards it as VXLAN encapsulated data packet based on the outer header. The packet is not sent to software on Spine 3. On Leaf 3, based on the appropriate loopback message signature, the packet will be sent to software VXLAN OAM module, which, in turn, will generate a loopback response that will be sent back to the originator Leaf 1.

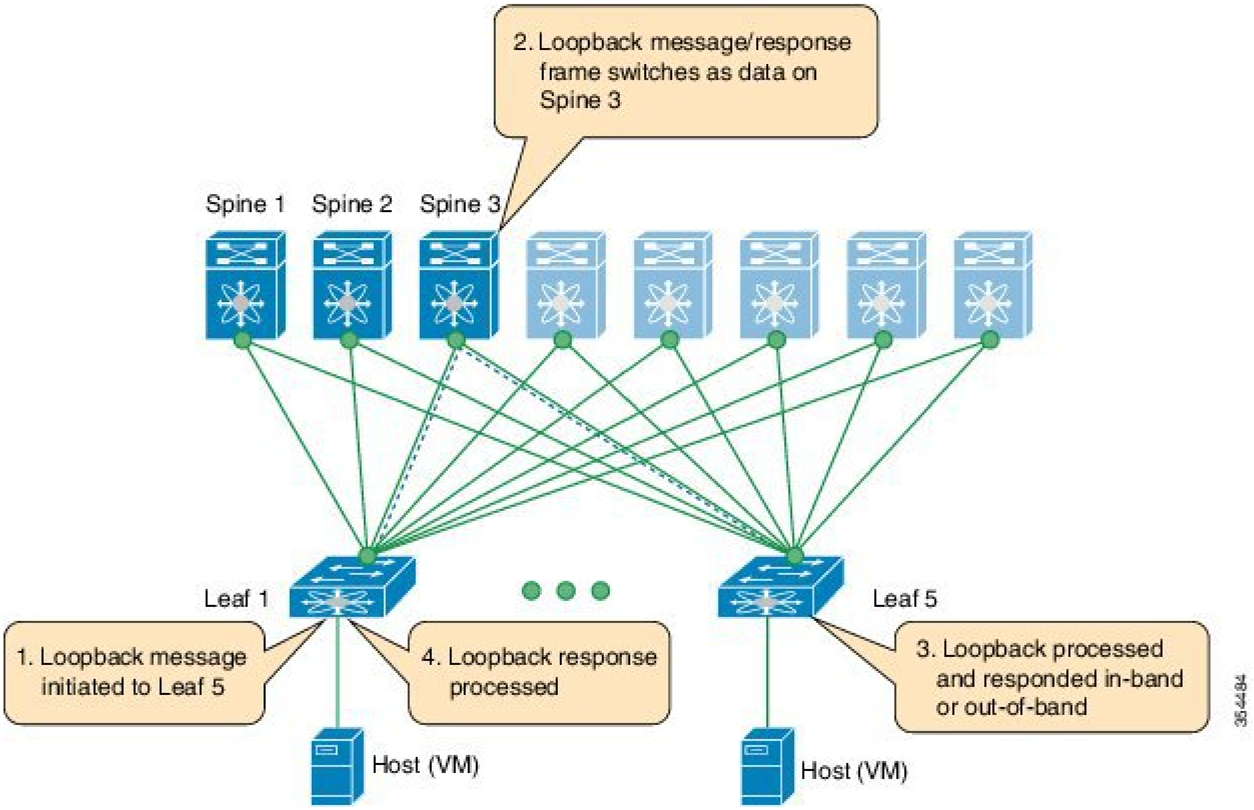
Here loopback (ping) message can be destined to VM or to the (VTEP on) leaf switch. This ping message can use different OAM channels. If the ICMP channel is used, the loopback message can reach all the way to the VM if the VMs IP address is specified. If NVO3 channel is used, this loopback message is terminated on the leaf switch attached to the VM, as the VMs do not support NVO3 headers in general. In that case leaf switch replies back to this message indicating the reachability of the VM. Ping message supports the following reachability options.

Ping

Check the network reachability (**Ping** command):

- From Leaf1 (VTEP1) to VM1 (attached host) (Only ICMP channel)
- From Leaf1 (VTEP1) to Leaf2 (VTEP2) (ICMP or NVO3 channel)
- From Leaf1 (VTEP1) to VM2 (host attached to another VTEP) (ICMP or NVO3 channel)

Figure 71: Loopback Message


2. Loopback message/response frame switches as data on Spine 3
Spine 1
Spine 2
Spine 3
Leaf 1
Leaf 5
1. Loopback message initiated to Leaf 5
4. Loopback response processed
3. Loopback processed and responded in-band or out-of-band
Host (VM)
Host (VM)
354484

Traceroute or Pathtrace Message

Traceroute or Pathtrace message is used for fault isolation. In a VXLAN network, it may be desirable to find the list of switches that are traversed by a frame to reach the destination. When the loopback test from a source switch to a destination switch fails, the next step is to find out the offending switch in the path. The operation of the path trace message begins with the source switch transmitting a VXLAN OAM frame with a TTL value of 1. The next hop switch receives this frame, decrements the TTL, and on finding that TTL is 0, it transmits a TTL expiry message to the sender switch. The sender switch records this message as an indication of success from the first hop switch. Then the source switch increases the TTL value by one in the next path trace message to find the second hop. At each new transmission, the sequence number in the message will be incremented. Each intermediate switch along the path decrements the TTL value by 1 as is the case with regular VXLAN forwarding.

This process continues until a response is received from the destination switch, or path trace process timeout occurs, or the hop count reaches a maximum configured value. The payload in the VXLAN OAM frames is referred to as the flow entropy. The flow entropy can be populated so as to choose a particular path among multiple ECMP paths between a source and destination switch The TTL expiry message may also be generated by intermediate switches for actual data frames. The same payload of the original path trace request is preserved for the payload of the response.

Traceroute and pathtrace are similar, except that traceroute uses ICMP channel, whereas pathtrace use NVO3 channel. Pathtrace uses NVO3 channel there by carries additional diagnostic information like interface load, statistics of the hops taken by these messages. If an intermediate device do not support NVO3 channel, this pathtrace behaves as simple traceroute and provides only the hop information.

Traceroute

Trace the path the packet traverses in the VXLAN overlay (**Traceroute** command):

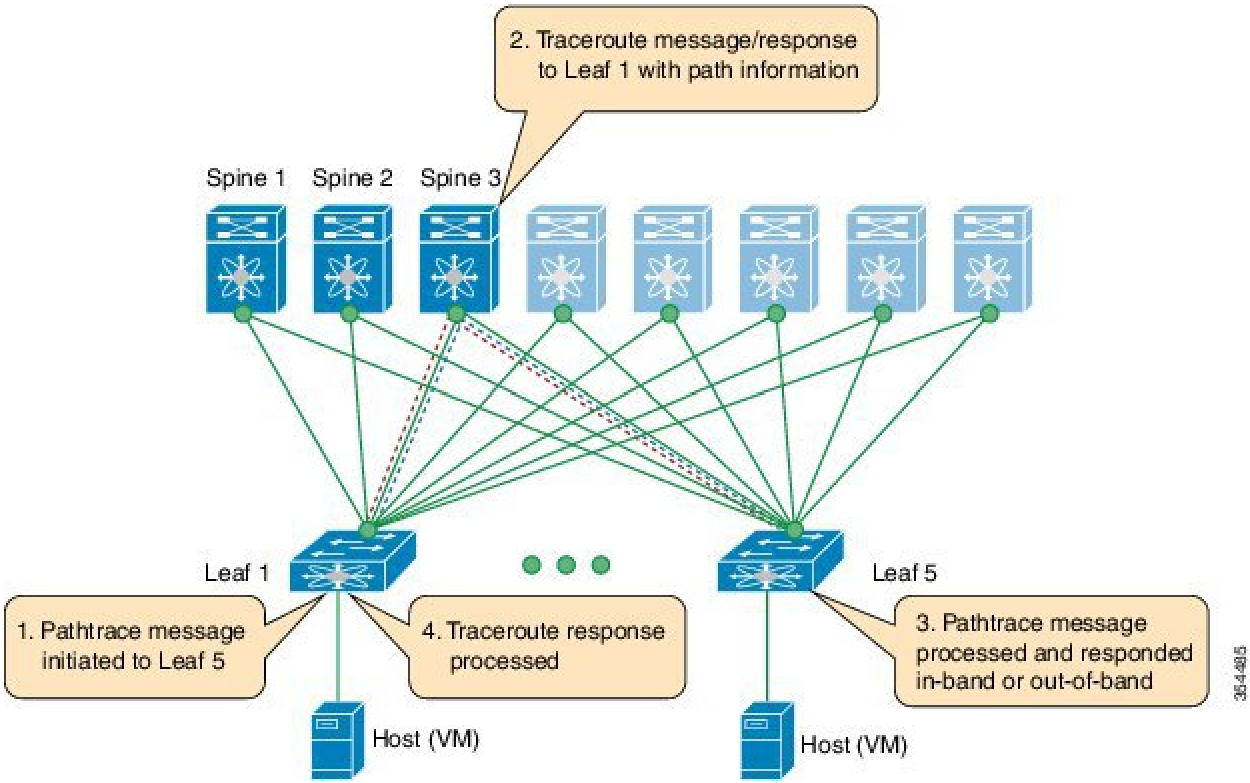
- Traceroute uses icmp packets (channel-1), encapsulated in the VXLAN encapsulation to reach the host

Pathtrace

Trace the path the packet traverses in the VXLAN overlay using the nvo3 channel (**Pathtrace** command):

- Pathtrace uses special control packets like NVO3 or TISSA (channel-2) to provide additional information regarding the path (for example, ingress interface and egress interface). But these packets will terminate at VTEP and does not reach the host, so only the VTEP will respond.

Figure 72: Traceroute Message


2. Traceroute message/response to Leaf 1 with path information
Spine 1
Spine 2
Spine 3
Leaf 1
Leaf 5
1. Pathtrace message initiated to Leaf 5
4. Traceroute response processed
3. Pathtrace message processed and responded in-band or out-of-band
Host (VM)
Host (VM)
354485

Configuring VXLAN OAM

As a prerequisite, ensure that the VXLAN configuration is complete. Create a loopback interface, assign an IP address with a /32 subnet mask, and associate the loopback interface with a customer VRF before verifying VXLAN OAM on the IP address. A sample configuration on two leaf switches is given below:

Leaf switch Leaf1

(config) #

```
interface loopback50
```

```
  vrf member VRF1
  ip address 209.165.201.2/32 tag 12345
```

Leaf switch Leaf2

(config) #

```
interface loopback50
  vrf member VRF1
  ip address 209.165.201.4/32 tag 12345
```

Step 1 Enters the NGOAM feature:

switch(config)# **feature ngoam**

Step 2 Install NGOAM access control list (ACL).

switch(config)# **ngoam install acl**

Example

The following example shows how to create ngoam profile from the configuration mode. You can create a profile to define various options like the dot1q, interface, source port, ip address, oam-channel, payload information and so on. After you define them, you can enter these pre-defined profiles names to execute certain operations instead of typing the respective commands.

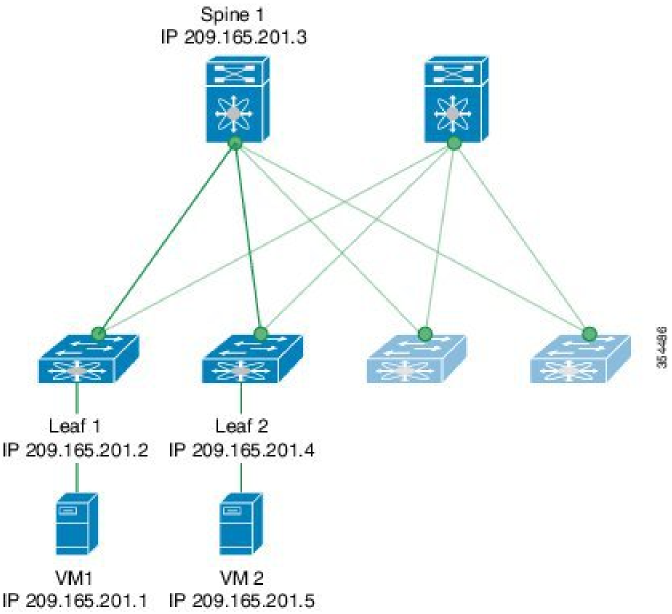
Note

- Channel 1 is enabled by default. So, no ngoam profile configuration is required for this channel.
- Channel 2 (NVO3) is not enabled by default, so **oam-channel 2** configuration is required under the **ngoam profile** command to use this channel.

(config) #

```
ngoam profile 1
  oam-channel 2
  sport 12345, 54321
  payload pad 0x2
  flow forward
    ip source 209.165.201.1
    ip destination 209.165.201.11

ngoam profile 2
  payload pad 0x3
  flow reverse
    ip source 209.165.201.2
    ip destination 209.165.201.12
```

Figure 73: VXLAN Network



VXLAN OAM provides the visibility of the host at the switch level, which allows a leaf to ping the host using the **ping nve** command.

The following example shows how to ping from Leaf 1 to the VM2, via Spine 1.

```
Switch# ping nve ip 209.165.201.5 vrf vni-31000 verbose

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
'D' - Destination Unreachable, 'X' - unknown return code,
'm' - malformed request(parameter problem),
'c' - Corrupted Data/Test

Sender handle: 34
! sport 40673 size 56,Reply from 209.165.201.5,time = 18 ms
! sport 40673 size 56,Reply from 209.165.201.5,time = 1 ms
! sport 40673 size 56,Reply from 209.165.201.5,time = 1 ms
! sport 40673 size 56,Reply from 209.165.201.5,time = 1 ms
! sport 40673 size 56,Reply from 209.165.201.5,time = 1 ms

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/18 ms
Total time elapsed 49 ms
```

The following example shows how to traceroute from Leaf 1 to the VM2 via Spine 1.

```
Switch# traceroute nve ip 209.165.201.5 vrf vni-31000 verbose

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
'D' - Destination Unreachable, 'X' - unknown return code,
'm' - malformed request (parameter problem),
'c' - Corrupted Data/Test

Traceroute request to peer ip 209.165.201.4 source ip 209.165.201.2
Sender handle: 36
  1 !Reply from 209.165.201.3,time = 1 ms
  2 !Reply from 209.165.201.4,time = 2 ms
  3 !Reply from 209.165.201.5,time = 1 ms
```

The following example shows how to pathtrace from Leaf 2 to Leaf 1.

```
Switch# pathtrace nve ip 209.165.201.4 vni 31000 verbose

Path trace Request to peer ip 209.165.201.4 source ip 209.165.201.2

Sender handle: 42
TTL   Code  Reply                IngressI/f    EgressI/f    State
======================================================================
1     !Reply from 209.165.201.3,  Eth5/5/1      Eth5/5/2     UP/UP
2     !Reply from 209.165.201.4,  Eth1/3        Unknown      UP/DOWN
```

Topology, Tenant and Overlay Provisioning

The Cisco Virtual Topology System (VTS) is a standards-based, open, overlay management and provisioning system for data center networks. It automates data center network fabric provisioning for virtual and physical infrastructure.

For information on Cisco VTS installation, see Cisco Virtual Topology System (VTS) 2.1 Installation Guide.

For information on VTS features, see Cisco Virtual Topology System (VTS) 2.1 User Guide.

CHAPTER 14

Cisco Nexus Fabric OpenStack Enabler Installation for the VXLAN BGP EVPN fabric

Hardware and Software Requirements

Cisco Nexus Fabric Enabler is a set of software applications that interacts with OpenStack through its open APIs to allow users to connect Cisco Nexus 5600, 6000, 7000 and 9000 Series platform switches as the network to the OpenStack compute nodes to form a cloud.

The uplink port of the server is directly connected to the fabric leaf switch and it requires that there is no middle device that has a bridging functionality, as LLDP needs to run between the server port and leaf switch port to signal the VM presence to the fabric for reachability. However, FEX devices can be connected between the server and the leaf switch as NX-OS 7.3 and above supports LLDP proxy running in the FEX.

OpenStack Juno release October 2014 or later, needs to be installed on any server using an OpenStack installer of your choice. This guide does not cover OpenStack installation.

- The qualification for this release is based on an OpenStack installation using DevStack.

- The qualification information of using other major third party OpenStack installers will be made available in the release notes document.

There are three ways to install OpenStack. They are given below:

- Use DevStack for installation. This is primarily for development and testing purposes and not for production level installation.
- Use a third-party supported OpenStack installation package. Nexus Fabric Enabler is officially qualified with Red Hat Enterprise Linux (RHEL) OSP 7.
- Use the OpenStack general installation guidelines.

In addition to OpenStack installation, the other required installation is LLDP Agent Daemon (lldpad), an open source free software, similar to OpenStack. The lldpad software is an agent daemon that supports VDP. For Linux distributions that packages lldpad using the source code after May 2015, follow the official installation procedure of the distribution to install lldpad. For example, in RHEL 7.2, install lldpad using **yum install lldpad**. Otherwise, follow the respective README notes to build and install the latest versions of lldpad on all servers used as compute nodes.

Note that, since OpenStack and its associated pieces of software for this purpose is open source software, it generally requires you to install the various software applications on your target servers as the applications are from different sources.

Cisco Nexus Fabric Enabler can be downloaded from the website https://github.com/CiscoSystems/fabric_enabler.

Supported Cisco Nexus Hardware and Software versions

Enabler Version	DCNM Version	Supported Platform	Cisco Nexus Switch Version	Fabric Type	OpenStack Release and Distribution
2.0	Cisco DCNM 10.1(1)	Nexus 9300	Cisco NX-OS 7.0(3)I5(2)	Cisco Programmable Fabric (PF) with VXLAN BGP EVPN	• Verified with RHEL OSP 8 (Liberty based). • Verified with Mitaka Release (devstack, Centos7.x, Ubuntu 14.04).
2.0	Cisco DCNM 10.1(1)	Nexus 5600	Cisco NX-OS 7.3(0)N1(1)	Cisco Programmable Fabric (PF) with VXLAN BGP EVPN Dynamic Fabric Automation (DFA)	• Verified with RHEL OSP 8 (Liberty based). • Verified with RHEL OSP 7 (Kilo based). • Verified with Mitaka Release (devstack, Centos7.x, Ubuntu 14.04).
1.1	Cisco DCNM 7.2(3)	Nexus 5600 Nexus 6000	Cisco NX-OS 7.1(0)N1(1)	DFA	Verified with RHEL OSP 6 (Juno based).

Note Cisco Nexus Fabric Enabler only supports the native DHCP server of OpenStack. For supporting any other external DHCP server, the DHCP server should have an IP Address Management (IPAM) plugin integrated with OpenStack. This is needed because, the IP address allocation module in OpenStack is a separate entity from the DHCP server and so it should be synchronized with the DHCP server. Usually, the IPAM plugins ensure that the DHCP server is synchronized with the IP address allocation logic in OpenStack.

Before you start

Following are the prerequisites for installing Nexus fabric OpenStack Enabler:

- Ensure that the fabric is up and running with the Power On Auto Provisioning (POAP) procedure being completed. This does not have any dependency with enabler installation or working, but this is a needed step for the overall functionality. POAP can be done even after the enabler installation is complete.
- Ensure that the interface on the switch connected to the server (uplink) has the following configuration:
  - **switchport mode trunk**
  - **encapsulation dynamic vdp**. This command is needed in switch versions NX-OS 7.3 and above.
- Ensure you know enough about OpenStack in general and have it installed and running in your setup (without the L3 service, or the Neutron router configured). OpenStack should be installed on the controllers and compute nodes.
- Ensure that the keystone and neutron services are running on the same server (the controller).
- Ensure each OpenStack compute node is connected to the leaf switch and the OpenStack control node is connected to Cisco DCNM through an IP network. Both the compute node and control node should be connected to the switches.
- Ensure that DCNM is installed and reachable from the controller node (OpenStack Controller) and from the Nexus Fabric cluster. Without this step, the enabler server process will gracefully exit.
- Ensure that all nodes have the same user account with the same credentials, and also that password-less sudo is enabled on all nodes.

  It can be checked in /etc/sudoers

  ```
  %sudo ALL=(ALL:ALL) NOPASSWD: ALL
  ```
- The RabbitMQ server must run on all the nodes. RabbitMQ is a general messaging scheme used by OpenStack for passing information.
- Ensure that lldpad is installed on all the controller and compute nodes. Refer the *Install lldpad* section for more details.

Install lldpad

On RHEL 7, lldpad can be installed by using the command **sudo yum install lldpad**.

On the other Linux distributions, ensure that their package manager is picking up lldpad after May, 2015. If not, the following needs to be done:

- Clone the repository used for installation—http://open-lldp.org/.
- Follow the instructions in the README file to build and install lldpad. You do not need the Linux kernel installation, but only the application.

Use the lldpad open source community mailing list http://open-lldp.org/ for any general queries.

Note All servers that will act as controller and compute nodes will need lldpad installation. The lldpad installation is also needed on the controllers because the solution uses the native DHCP server provided by OpenStack, and the DHCP virtual interface also needs to be auto-configured in the attached leaf switch.

Note The OpenStack DHCP service is supported while the Cisco DCNM DHCP service is not supported.

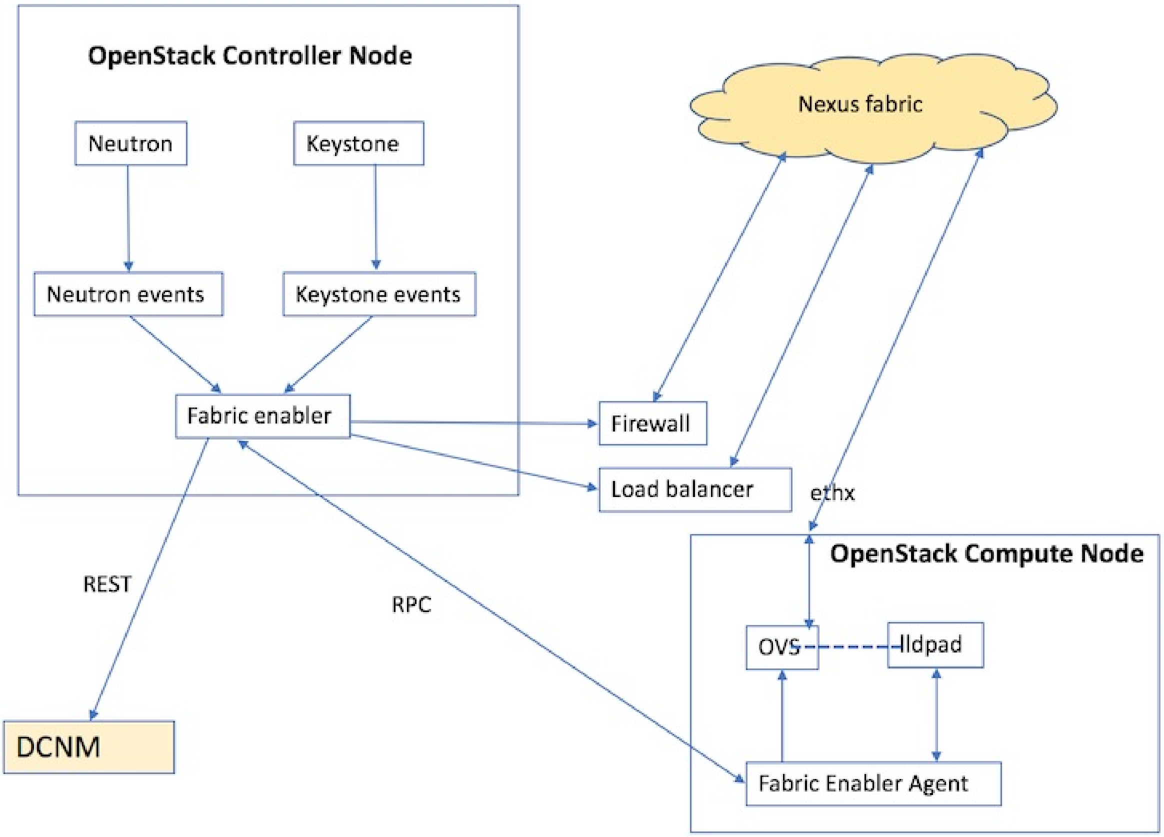
Assumptions

- You have an understanding of Cisco Nexus Fabric system architecture. For more details, see the Deploy a VXLAN Network with an MP-BGP EVPN Control Plane white paper.
- You have an understanding of OpenStack. This document does not explain the functionality of Openstack and its components.

Architecture

The architecture and the different components of the Nexus Fabric Enabler solution is shown below:

Figure 74: Nexus Fabric Enabler architecture



Nexus Fabric Enabler is a solution to integrate OpenStack with the Cisco Nexus VXLAN BGP EVPN fabric using DCNM as the controller. In other words, it offers the Programmable (VXLAN BGP EVPN) Fabric or the Dynamic Fabric Automation (DFA) solution with OpenStack as the orchestrator. The solution has three key components:

1. Nexus Fabric Enabler server.
2. Nexus Fabric Enabler agent.
3. LLDP Agent Daemon (lldpad).

The components are described below. This solution is not a plugin that runs in the context of OpenStack. The Nexus Fabric Enabler solution listens to OpenStack notifications and acts on it to achieve the functionality. In other words, Nexus Fabric Enabler is loosely coupled with OpenStack.

Nexus Fabric Enabler Server

Nexus Fabric Enabler server functions are given below:

- The server runs as a service in the OpenStack Controller node. Both Systemd and Upstart scripts are provided to manage the service.
- In case of HA setups, the server runs on only one controller. In case of RHEL OSP 7 or 8, the enabler server can be managed by PaceMaker (using **pcs** CLIs), which can decide on the controller to launch the Fabric Enabler server.

- The server listens to event notifications from OpenStack's Keystone and Neutron services through RabbitMQ.
- The server communicates project and network events to DCNM. A project create event in OpenStack translates to an organization create in DCNM along with a partition create. Every project in OpenStack maps to an organization/partition in DCNM. Similarly, a network create in OpenStack translates to a network create in DCNM under the partition, which was created during project create.
- The server manages segmentation for tenant networks. Every network created in OpenStack is assigned a unique segment ID.
- The server communicates VM events to the appropriate Nexus Enabler agent. When a VM is launched through OpenStack, the Nexus Enabler server also gets the compute node that is going to host the VM. The VM information, along with the relevant project and network details, are communicated to the Nexus Enabler agent running in the compute node.
- Database management—The MySQL database is used for persistency so that the state is not lost between process restarts.
- Error handling—The Nexus enabler server has to ensure that DCNM and OpenStack remain in sync. If a project or network creation/deletion fails in DCNM temporarily, the enabler server has to retry the operation. Similarly, auto-configuration for a VM can also fail in the switch temporarily, in which case the enabler server needs to ensure that the enabler agent retries the operation.
- The server is completely restartable.

Nexus Fabric Enabler Agent

Fabric Enabler agent functions are given below:

- The Fabric Enabler agent runs as a service in all the OpenStack (controller and compute) nodes. Both Systemd and Upstart scripts are provided to manage the service.
- The agent performs uplink detection and programs OVS flows to pass through LLDP/VDP packets, through the OVS bridge. This is a feature by which the Fabric Enabler agent dynamically determines which interface of the compute node is connected to the ToR switch. This interface is referred as the *uplink* interface.

    - Unlike many deployment scenarios in OpenStack, it is not necessary to provide the uplink interface of the server during OpenStack installation. The interface information need not be consistent in all the servers. One compute server can have 'eth2' as the uplink interface and another server can have 'eth5' as the uplink interface.

    - This function works for port-channel (bond) uplink interfaces.

    - This works only when the uplink interface is not configured with an IP address.

    - When more than one interface is connected to the ToR switch, and if the interfaces are not a part of a port-channel (bond), then the uplink detection will pick the first available interface. For special scenarios, such as this one, it is recommended to turn off uplink detection and provide the uplink interface for this compute node in the configuration file as explained in the next chapter.

- The agent communicates vNIC information of a VM along with Segment ID to VDP (running in the context of lldpad) after being notified by the Fabric Enabler server. This communication is done periodically. It is needed for failure cases because lldpad is stateless for vNICs for which auto-configuration has failed. So, the vNIC information needs to be periodically refreshed to lldpad. This is also needed during lldpad restarts, because lldpad does not store the vNIC information in a persistent memory.
- The agent retrieves all the VMs running on this compute node from the Fabric Enabler server at the beginning. The Fabric Enabler agents do not have a persistent DB (like MySQL). It relies on the server to maintain the information. This is similar to OpenStack's Neutron server and agent component.

- The agent programs the OVS flows based on the VLAN received from the switch through VDP.
- The agent is completely restartable.

LLDP Agent Daemon (lldpad)

Pointers about lldpad are given below:

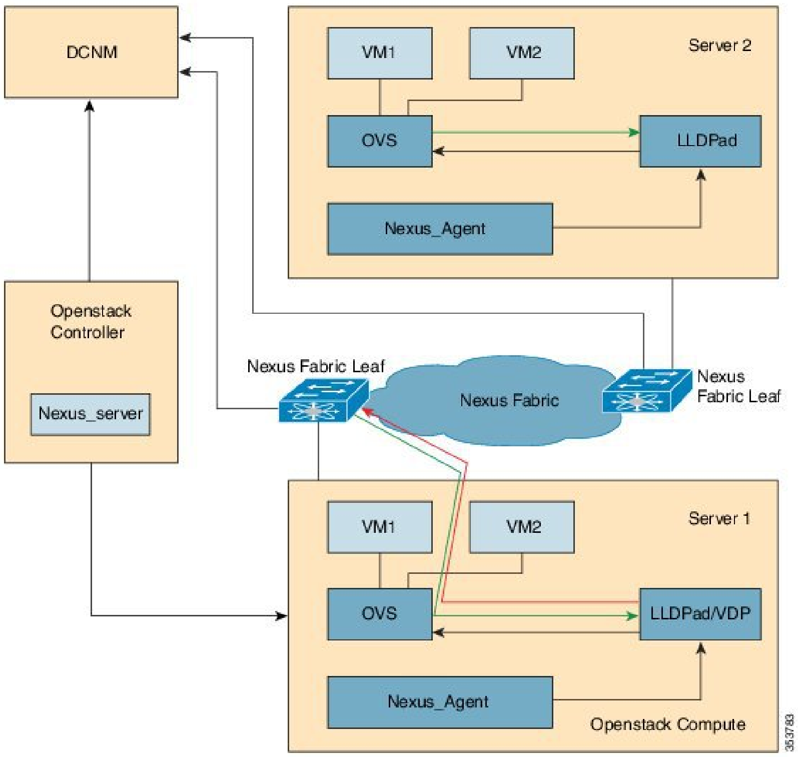
- It is an open source software supporting VDP. Refer to the IEEE 802.1QBG document for more details on VDP functionality.

Cisco Nexus Fabric Enabler System Flow

OpenStack serves as one of orchestrators of the cloud enabled through the Cisco Nexus fabric. For this release, all the orchestration is done using OpenStack's *Horizon* dashboard graphical user interface, CLIs, or OpenStack generic open APIs.

The following diagram provides a high level overview of the OpenStack orchestrator with the Cisco Nexus fabric as its network connecting all the compute nodes:

Figure 75: System Flow



The control flow can be summarized as follows:

1. OpenStack cloud administrator creates a tenant or project. OpenStack Keystone component notifies the Cisco Nexus Fabric Enabler server running in the control node about the creation of the tenant. The server sends the tenant information to Cisco Prime DCNM through the Cisco Prime DCNM REST API. The tenant's user name and password are created by the administrator too.

2. The tenant logs into OpenStack and creates the network. The OpenStack Neutron components notifies the Cisco Nexus Fabric Enabler server of the network Information (subnet/mask, tenant name), and the server automatically sends the network creation request to Cisco Prime DCNM through the Cisco Prime DCNM REST APIs.
3. A VM instance is launched, specifying the network that the instance will be part of.
4. The server sends the network information (subnet/mask, tenant name) and VM information to the Cisco Nexus Fabric Enabler agent running in the corresponding compute node.
5. VSI Discovery and configuration protocol (VDP), running in the context of lldpad, gets notified by the agent about the VM and the segment ID associated with the VM.
6. VDP, in turn, communicates with the leaf switch and sends the VM's information with the segment ID.
7. The leaf switch contacts Cisco Prime DCNM with the segment ID for retrieving the network attributes and auto-configure the compute node facing the interface for this tenant VRF.
8. The leaf switch responds with the VLAN information that is to be used for tagging the VM's traffic. The VLAN to be used will be the first available VLAN from the predefined VLAN pool (created using the **system fabric dynamic-vlans** command) on each individual leaf switch. The selected one is the leaf switch significance resource.
9. lldpad module notifies the Cisco Nexus Fabric Enabler agent with the VLAN information provided by the leaf switch.
10. The agent configures OVS such that the VM traffic to the network contains the VLAN information/tag provided by the leaf switch. The VM's vNIC is operational only at this point.

Note **Attention**—Refer the prerequisites for installing Nexus fabric OpenStack Enabler. It is noted in the *Before You Start* section of the *Hardware and Software Requirements* chapter.

Installation Overview

Apart from OpenStack installation, the Nexus Fabric Enabler also needs to be installed. As described previously, OpenStack can be installed in multiple ways. Apart from installation, the OpenStack configuration file needs to have certain settings for the solution to work. The changes in the configuration file are needed for the following:

- Configuring OpenStack networking to work in conjunction with the VXLAN BGP EVPN farbic (Programmable Fabric) or DFA. This is done when OpenStack is installed for the first time. This step is dependent on the distribution. For example, RHEL OSP 7 may have a different command to perform this operation than Mirantis.
- Enabling notification functionality in OpenStack, which can be received by the Nexus Fabric Enabler. This step is done by the installer provided by the Nexus Fabric Enabler. This step is mostly common across distributions.

The Nexus Fabric Enabler solution is officially qualified with RHEL OSP 7. The solution is also tested with DevStack, which is not intended for production. This chapter starts with a section on Red Hat OSP installation that provides pointers on *configuring OpenStack networking to work in conjunction with Programmable Fabric or DFA*. Then, the installation mechanism of Nexus Fabric Enabler is explained in detail.

OpenStack Installation

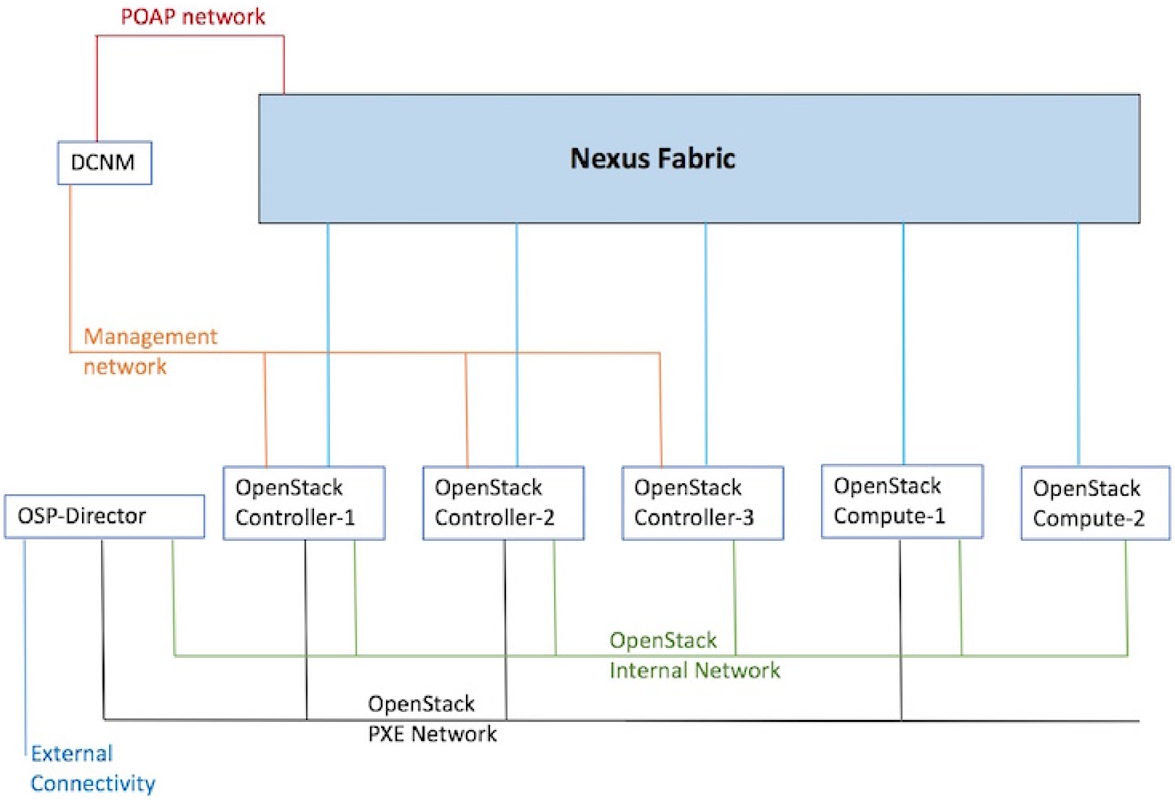
Topology

Note

Important—A sample topology is shown below. This is a critical step as it lays down the foundation for installation. It is highly recommended that you follow the same wiring scheme. This topology is based on RHEL OSP 7/8 with the Nexus fabric, Fabric Enabler, and DCNM. In the topology, the interface that is connected to the cluster should be operationally up (the corresponding command is **sudo ifconfig eth0 up**).

DCNM is also connected to the OpenStack control node through an IP network. Refer the OSP installation guide for more information on the OpenStack internal and PXE network.

Figure 76: Sample topology



RHEL OSP 7 Installation

Note Refer the official Red Hat documentation on how to install the Overcloud and the Undercloud.

Before deploying the Overcloud, implement the following steps to ensure compatibility with Nexus Fabric Enabler.

1. The Neutron type drivers must include *local*. Add these Neutron Type Drivers in *network-environment.yaml*, the network environment file—*local*, *flat*, *vlan*, *gre*, and *vxlan*.
2. Disable Neutron tunneling with the -- *neutron-disable-tunneling* option.
3. Set the Neutron Network Type to *local* with the -- *neutron-disable-tunneling* option.
4. Set the Neutron bridge mapping with the -- *neutron-bridge-mappings ethd:br-ethd* option.

An example for an Overcloud deployment command that is compatible with Nexus Fabric Enabler is given below:

```
openstack overcloud deploy --templates \

-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /home/stack/templates/network-environment.yaml \
-e /home/stack/templates/storage-environment.yaml \
--control-flavor control \
--compute-flavor compute \
--ntp-server clock.cisco.com \
--neutron-network-type local \
--neutron-disable-tunneling \
--neutron-bridge-mappings ethd:br-ethd \
--compute-scale 2 \
--verbose
```

HA for RHEL OSP 7

High availability (HA) provides continuous operation. The RHEL OSP7 director provides high availability to an OpenStack Platform environment through the controller node cluster. The director installs a set of the same components on each controller node and manages them as one whole service. Having a cluster provides a fallback in case of operational failures on a single controller node.

The following example shows how to install the Overcloud with redundant controllers:

```
openstack overcloud deploy --templates \

-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /home/stack/templates/network-environment.yaml \
-e /home/stack/templates/storage-environment.yaml \
--control-flavor control \
--compute-flavor compute \
--ntp-server clock.cisco.com \
--neutron-network-type local \
--neutron-disable-tunneling \
--neutron-bridge-mappings ethd:br-ethd \
--control-scale 3 \
--compute-scale 2 \
--verbose
```

Note For HA to work, you need a minimum of three controllers.

Note If compute nodes are connected to fabric leaf nodes through a port-channel/bond, make sure the Linux bond interfaces are used, since OVS bond interfaces will not work with Fabric Enabler. Create bond interfaces before installing and running the Openstack Fabric Enabler.

Verification of Configuration Files

This section is generally independent of the OpenStack installation method. But, check with the distribution as to where the configuration files are placed, and if the name of the configuration file has changed. As described earlier, the type_drivers should be set to local. *openvswitch* is used as the vswitch for the VXLAN BGP EVPN Programmable Fabric or DFA solution. Verify the following:

1. Ensure that *./etc/neutron/plugins/ml2/ml2_conf.ini* is configured as follows:

   ```
   [ml2]
   type_drivers = local mechanism_drivers = openvswitch

   [ovs]
   bridge_mappings = ethd:br-ethd

   ([ml2_type_flat], [ml2_type_vlan], [ml2_type_gre] and [ml2_type_vxlan] sections should
   not be specified)
   ```

2. Ensure that *./etc/neutron/neutron.conf* is configured as follows:

   ```
   core_plugin = neutron.plugins.ml2.plugin.Ml2Plugin
   ```

   Ensure that the tunnel type is not set. The *keystone_authtoken* should be similar to the following setting:

   ```
   [keystone_authtoken]
   signing_dir = /var/cache/neutron
   auth_uri = http://<ip address of controller>:5000/v2.0
   cafile = /opt/stack/data/ca-bundle.pem
   identity_uri = http://<ip address of controller>:35357
   auth_host = <ip address of controller>
   auth_port = 35357
   auth_protocol = http
   admin_tenant_name = service
   admin_user = neutron
   admin_password = password
   ```

3. Ensure that *./etc/nova/ nova.conf* has the *keystone_authtoken* section(s) set similar to the following

   ```
   [keystone_authtoken]
   signing_dir = /var/cache/nova
   ```

```
admin_password = password
admin_user = nova
admin_tenant_name = service
auth_uri = http://<ip address of controller>:5000/v2.0
cafile = /opt/stack/data/ca-bundle.pem
identity_uri = http://<ip address of controller>:35357
auth_protocol = http
auth_port = 35357
auth_host = <ip address of controller>
```

Note Different OpenStack distributions may have the configuration file placed in a different directory. Or, the name of the configuration file can be different. Ensure that the appropriate configuration file has the above contents set correctly.

Cisco Nexus Fabric Enabler Installation

The sections below provides Cisco Nexus Fabric Enabler installation information.

Prerequisites

The following pre-requisite is applicable for RHEL OSP 7/8 based setups.

Note All the changes to Neutron and Nova files as well as extra rpm installation, Neutron DB patching for the firewall, disabling selinux, etc, that are described for a non-HA setup are still applicable.

- Cisco Nexus Fabric Enabler requires RabbitMQ to listen to requests made to the VIP address. You should configure this manually as other OpenStack components do not need them.

Installation in all the nodes (Fresh install)

1. Login to the OSP 7 Director Node for an RHEL OSP 7 or OSP 8 setup. In case of a non-production setup like DevStack, login to the controller node.
2. git clone -b rel_2_0_0 https://github.com/CiscoSystems/fabric_enabler ofe
3. cd ofe
4. Edit enabler_conf.ini as given in the next section.
5. The following command will install the Fabric Enabler on all the controller and compute nodes. This command will also install the Fabric Enabler in case of a HA setup. The two scenarios involving proxy requirement are given below:
   - If a proxy for reaching the Internet is not needed, use these commands:

     ```
     python setup_enabler.py  --vendor-os-release=rhel-osp7 --mysql-user=root
     --mysql-host=localhost
     ```

   - If a proxy for reaching the Internet is needed, use these commands:

     ```
     python setup_enabler.py  --vendor-os-release=rhel-osp7 --mysql-user=root
     --mysql-host=localhost --https-proxy=<proxy>
     ```

<proxy> is the https proxy, such as *https://proxy.esl.cisco.com:80* from Cisco Labs.

In case of HA setups for RHEL OSP 7 or OSP 8, the Enabler server will be started by Pacemaker. The Enabler server will be running in one of the servers. In case of a crash or the controller node going down, Pacemaker will ensure to restart the Enabler server in one of the available controllers. The Enabler agent and lldpad will be started in all the HA controller nodes and compute nodes.

Installation on a single node

Typically, this is needed when a new compute node is added after installing the Fabric Enabler on all the nodes:

1. Login to the OSP7 Director Node in case of RHEL OSP 7 or OSP 8 setups. In case of a non-production setup like DevStack, login to the controller node.
2. git clone -b rel_2_0_0 https://github.com/CiscoSystems/fabric_enabler ofe
3. cd ofe
4. Edit enabler_conf.ini as given in the next section.
5. Use the Nova list to get a list of controllers and compute node IP addresses.
6. For each compute node that is newly added, install the Fabric Enabler using these commands:

```
python setup_enabler.py --compute-name=<compute ip address> --remote-user=heat-admin
--https-proxy=<proxy>
```

Upgrading Fabric Enabler to the new version

The following are needed when you want to install a new version of the Fabric Enabler:

1. Ensure proxy variables are set properly.
2. Update the Fabric Enabler git repository using the following command:

```
git pull
```

Note If the git repository is not available, a new one needs to be cloned.

3. Ensure *enabler_conf.ini* is up to date. If needed, copy it from a controller.
4. If a proxy for reaching the internet is not needed, use these commands:

```
python setup_enabler.py  --vendor-os-release=rhel-osp7 --upgrade=True
```

5. If a proxy for reaching the internet is needed, use these commands::

```
python setup_enabler.py  --vendor-os-release=rhel-osp7 --https-proxy=<proxy>
--upgrade=True
```

   a. More upgrade options/pointers are given below:
      - The upgrade process will upgrade and restart the Cisco Nexus Fabric Enabler server and agent.

- At times, lldpad also needs to be restarted. If so, add *--restart-lldpad=True* to the above commands.
- It is possible to update a single controller or compute note by using the *--control-name* and *--compute-name* options, respectively.

enabler_conf.ini

The following table describes the sections and fields in the *enabler_conf.in* file.

Note Unless specified as *(Optional)* in the Field Name column, the field is mandatory.

Table 8: Section—[General]

Field Name	Description	Default Value	Fabric Type
compute_user (Optional)	Compute node user name that can be used with the **ssh** command for remote logins. Also, the user must be a sudoer assuming all compute nodes have the same	Not Applicable	VXLAN BGP EVPN (Programmable Fabric) DFA
compute_passwd (Optional)	Compute node password that can be used with the **ssh** command for remote logins.	Not Applicable	VXLAN BGP EVPN DFA
node (Optional)	A *comma* separated list of hosts for which a static uplink is configured. The node name should be a fully qualified domain name (such as **host1.example.com**)	Not Applicable	VXLAN BGP EVPN DFA
node_uplink (Optional)	A comma separated list of uplink ports on the server connected to the leaf switch. This parameter and the *node* parameter are mandatory if a static uplink is desired.	Not Applicable	VXLAN BGP EVPN DFA

ucs_fi_evb_dmac (Optional)	If OpenStack is running in any UCS FI blade server, enter the EVB DMAC address that is configured in the fabric. The Fabric Enabler software running in the node will detect if it is a UCS FI blade server, but the interface connected to the switch has be included in the 'node_uplink' configuration, along with the node.	01:80:c2:12:34:56	VXLAN BGP EVPN DFA

This is the section about Cisco DCNM, which you have installed separately and set the right access credentials to be used by the OpenStack Cisco Nexus Fabric Enabler. Ensure that the *gateway_mac* value matches your POAP template setting in Cisco DCNM for your leaf switch, and you use the right range of segment IDs administrated by your Fabric Manager.

Table 9: Section—[dcnm]

Field Name	Description	Default Value	Fabric Type
dcnm_ip	IP address of the DCNM. It should be reachable from the OpenStack controller node.	Not Applicable	VXLAN BGP EVPN DFA
dcnm_user	DCNM server login credentials.	Not Applicable	VXLAN BGP EVPN DFA
dcnm_amqp_user	DCNM server RabbitMQ messaging credentials.	Not Applicable	VXLAN BGP EVPN DFA
dcnm_password	DCNM server password.	Not Applicable	VXLAN BGP EVPN DFA
gateway_mac (Optional)	Gateway MAC address. This should be the same as the MAC address configured on the leaf switch nodes.	20:20:00:00:00:AA	VXLAN BGP EVPN DFA
orchestrator_id (Optional)	Orchestartor ID used for registering the segment ID range on DCNM. If there are multiple setups using the same DCNM, ensure different orchestrator IDs are used.	Openstack Controller	VXLAN BGP EVPN DFA

segmentation_id_min	The minimum Segment ID value. It is a 24 bit integer value.	4097	VXLAN BGP EVPN DFA
segmentation_id_max	The maximum Segment ID value. It is a 24 bit integer value.	16777216	VXLAN BGP EVPN DFA
segmentation_reuse_timeout (Optional)	Duration after which an *available* segment ID can be reused. Once a segment ID is released, it will only be reused after 1 hour. If this functionality is not needed, enter a value of 0. Alternatively, to change the default value of 1 hour, uncomment the below and enter a different number (as an integer value).	1 hour	VXLAN BGP EVPN DFA
dcnm_net_ext (Optional)	The suffix of a network name when it is created by DCNM. This is usable for a scenario when network creation is done in DCNM and the Fabric Enabler populates this in OpenStack.	(DCNM)	VXLAN BGP EVPN DFA
dcnm_dhcp_leases (Optional)	The lease file name of the DHCP server on the DCNM.	/var/lib/dhcpd/dhcpd.leases	VXLAN BGP EVPN DFA
default_cfg_profile (Optional)	Default configuration profile when creating a network in DCNM.	defaultNetworkEvpnProfile defaultNetworkUniversalTfProfile	VXLAN BGP EVPN DFA
default_vrf_profile (Optional)	Default VRF profile name for a partition in DCNM.	vrf-common-evpn vrf-common-default	VXLAN BGP EVPN DFA

Table 10: Section—[dfa_rpc]

Field Name	**Description**	**Default Value**	**Fabric Type**

transport_url	Transport URL parameter for RPC.	Not Applicable An example is given below: `transport_url='rabbit://username:password@%(ip)s:5672//'` The *ip* address is of the controller when there is only one controller. In a HA environment with multiple controllers in a cluster, the IP address is the virtual IP address (VIP) of the controller cluster. You should replace the *username* and *password* based on your setting. These credentials should be the same as the one you used to configure RabbitMQ, by default available in the location */etc/rabbitmq/rabbitmq.config*.	VXLAN BGP EVPN DFA

Table 11: Section—[dfa_mysql]

Field Name	**Description**	**Default Value**	**Fabric Type**
connection	MYSQL DB connection option	Not Applicable An example is given below: `connection=mysql://username:password@localhost/cisco_dfa?charset=utf8` The *localhost* is applicable if there is only one controller. In a HA environment with multiple controllers in a cluster, a localhost will be replaced with the *VIP* of the controller cluster. You should replace the *username* and *password* based on your setting.	VXLAN BGP EVPN DFA

Table 12: Section—[dfa_notify]

Field Name	**Description**	**Default Value**	**Fabric Type**
cisco_dfa_notify_queue (Optional)	Notification queue name for DFA enabler. service_name: keystone and neutron.	cisco_dfa_%(service_name)s_notify	VXLAN BGP EVPN DFA

Table 13: Section—[dfa_log]

Field Name	**Description**	**Default Value**	**Fabric Type**

log_file (Optional)	Log file name. DEPRECATED (use Log file prefix instead). If log file name and directory is not specified, the default is the standard output.	fabric_enabler.log	VXLAN BGP EVPN DFA
log_file_prefix (Optional)	The prefix will be used by Fabric Enabler processes to create log files. If the default prefix of fabric_enabler is used, the Fabric Enabler server's log files will be fabric_enabler_server.log and the Fabric Enabler agent's log file will be fabric_enabler_agent.log	fabric_enabler	VXLAN BGP EVPN DFA
log_dir (Optional)	The directory name for the log file.	Current directory	VXLAN BGP EVPN DFA
log_level (Optional)	Enabler debugging output level. Set to DEBUG to see the debbugging output	WARNING	VXLAN BGP EVPN DFA

Table 14: Section—[dfa_agent]

Field Name	Description	Default Value	Fabric Type
integration_bridge (Optional)	OVS Neutron Agent related configuration. Ensure that this is the same as what is configured for the OVS Neutron Agent.	br-int	VXLAN BGP EVPN DFA
external_dfa_bridge (Optional)	OVS Neutron Agent related configuration. Ensure that this is the same as what is configured for the OVS Neutron Agent.	br-ethd	VXLAN BGP EVPN DFA

Table 15: Section—[vdp]

Field Name	Description	Default Value	Fabric Type
mgrid2 (Optional)	Refer to IEEE 801.1QBG standard documentation.	0	VXLAN BGP EVPN DFA
typeid (Optional)	Refer to IEEE 801.1QBG standard documentation.	0	VXLAN BGP EVPN DFA

typeidver (Optional)	Refer to IEEE 801.1QBG standard documentation.	0	VXLAN BGP EVPN DFA
vsiidfrmt (Optional)	Refer to IEEE 801.1QBG standard documentation.	5	VXLAN BGP EVPN DFA
hints (Optional)	Refer to IEEE 801.1QBG standard documentation.	Not Applicable	VXLAN BGP EVPN DFA
filter (Optional)	Refer to IEEE 801.1QBG standard documentation.	4	VXLAN BGP EVPN DFA
vdp_sync_timeout (Optional)	Query to lldpad for every VSI. If a VSI is not present in lldpad, an associate request is sent to lldpad.	15	VXLAN BGP EVPN DFA

Note

- Ensure that the segment ID does not overlap with other segment ID ranges (for example, the Cisco DCNM segment ID uses the default 30,000 to 49,999).
- It requires all control and compute nodes that have the same username and password, and this is your Linux account on the servers, to run as control/compute nodes.

Post Installation

Verify the Cisco Nexus Fabric Enabler server, Cisco Nexus Fabric Enabler agent, lldpad, and the existence of notification queues, as shown below:

Cisco Nexus Fabric Enabler server verification

On an RHEL OSP HA setup, a sample output is shown below on a controller:

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs resource | grep fabric-enabler-server
 fabric-enabler-server  (systemd:fabric-enabler-server):       Started overcloud-controller-1
```

The command displays the controller node where the Fabric Enabler server is running. Alternatively, the following commands can be used on the controller where the Fabric Enabler server is running:

- sudo systemctl status fabric-enabler-server [For a Redhat/CentOs based controller]
- sudo status fabric-enabler-server [For a Ubuntu based one]
- ps –ef | grep fabric-enabler-server [Any setup]

Cisco Nexus Fabric Enabler agent verification

The Enabler agent runs on all controller and compute nodes. Run the below commands on a node where verification is needed. This sample is specific to Red Hat based setups that have system based startup scripts.

```
[heat-admin@overcloud-controller-0 ~]$ sudo systemctl status fabric-enabler-agent
â fabric-enabler-agent.service - Cluster Controlled fabric-enabler-agent
   Loaded: loaded (/usr/lib/systemd/system/fabric-enabler-agent.service; enabled; vendor
preset: disabled)
  Drop-In: /run/systemd/system/fabric-enabler-agent.service.d
           ââ50-pacemaker.conf
   Active: active (running) since Fri 2016-09-23 01:31:01 EDT; 1 weeks 0 days ago
```

Alternatively, the following commands can be used:

- sudo status fabric-enabler-agent [On Ubuntu based servers]
- ps –ef | grep fabric-enabler-agent [Any setup]

lldpad verification

lldpad runs on all controller and compute nodes. Run the below commands on the compute node where verification is needed. The sample is specific to Red Hat based setups that have system based startup scripts.

```
[heat-admin@overcloud-controller-0 ~]$ sudo systemctl status lldpad
â lldpad.service - Cluster Controlled lldpad
   Loaded: loaded (/usr/lib/systemd/system/lldpad.service; enabled; vendor preset: disabled)

  Drop-In: /run/systemd/system/lldpad.service.d
           ââ50-pacemaker.conf
   Active: active (running) since Sun 2016-08-07 22:05:27 EDT; 1 months 22 days ago
 Main PID: 6041 (lldpad)
   CGroup: /system.slice/lldpad.service
           ââ6041 /usr/sbin/lldpad –t
```

Alternatively, the following commands can be used:

- sudo status lldpad [On Ubuntu based servers]
- ps –ef | grep lldpad [Any setups]

Verify existence of notification queues

```
sudo rabbitmqctl list_queues | grep cisco cisco_dfa_keystone_notify.info 0
cisco_dfa_neutron_notify.info 0
```

Create Project and Launch VM

The information provided in this section is generic to OpenStack and it is provided here for your convenience with the exception of ConfigProfile, which is Cisco Nexus fabric specific.

Steps to Create a Project

Follow these steps to create a project:

1. Login to the Horizon dashboard as an administrator. Use the password that you used in the OpenStack configuration file.
2. Click **Projects** and then **Create Project**.
3. Enter relevant project information and click **Create Project** to create the project.

Note The project name is used as vrfName in the fabric (vrfName = “project_name:CTX”) for fabric auto-configuration. The fabric limits the size of the vrfName string to 32 characters. Ensure that the project name length is less than 29 characters when creating the project. Do not use hyphens in the project name.

DCI Support

You can use OpenStack to configure the DC Inter-connect function. Support is only provided for Layer-3 DCI with the Cisco Prime DCNM 7.1(1) release, and Cisco NX-OS 7.1(0)N1(1) release or later.

As part of the project name string, type *xyx:dci_id:129* to enable DCI support ('129' is used here as an example). Type *xyz* or *xyz:dci_id:0* to remove DCI support for this project.

The integer 129 is the DCI ID. Cisco Prime DCNM uses it as an indication that the user desires to auto-configure the border leaf switches with this VRF, and extend to the DCI edge devices(s). If the value is *0*, Cisco Prime DCNM removes VRF configurations from the border leaf switches and the configurations that extend the VRF from the border leaf switch to the DC edge device(s).

Steps to Create a User for the Project

Follow these steps to create a user for the project:

1. Click **Users**, and then **Create User**.
2. Fill in all the fields, select the project you just created and select the role as *admin*. The network information will not be populated correctly to DCNM if you fail to specify the role as *admin*.

Steps to Create the Network

Follow these steps to create the network:

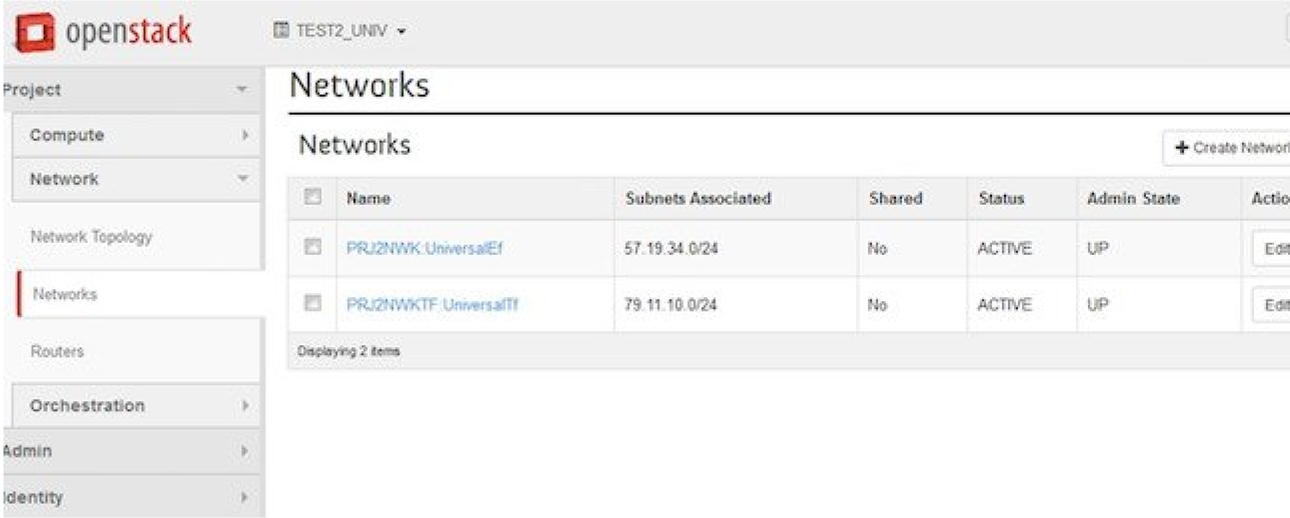
1. Login as a user using login credentials created by the administrator.
2. Click the **Project** tab.
3. Click **Networks** and then click **Create Network**. Specify a Name for the network and go to the **subnet** tab. This is mandatory.

4. Specify a Network Address for the subnet.

Use non-default network profiles

By default, for Cisco Prime DCNM with version 7.1, defaultNetworkUniversalEfProfile is the network profile used automatically by the system. Additionally, defaultNetworkUniversalTfProfile can also be specified when creating a network in OpenStack. A sample screen shot is given below:

Figure 77: Default Network



□	Name	Subnets Associated	Shared	Status	Admin State	Actio
□	PRJ2NWK:UniversalEf	57.19.34.0/24	No	ACTIVE	UP	Edi
□	PRJ2NWKTF:UniversalTf	79.11.10.0/24	No	ACTIVE	UP	Edi

Displaying 2 items

Following are the supported network profiles with Cisco Prime DCNM version 7.1(1):

- defaultNetworkUniversalEfProfile
- defaultNetworkUniversalTfProfile
- defaultNetworkL2Profile

If it is an upgrade from version 7.0(1) or 7.0(2) to 7.1(1), the default profile will be defaultNetworkIpv4EfProfile, and the supported profiles will be the sum of the profiles for versions 7.0(1), 7.0(2) and 7.1(1) or later, as shown below:

- defaultNetworkIpv4EfProfile
- defaultNetworkIpv4TfProfile
- defaultNetworkL2Profile
- defaultNetworkUniversalEfProfile
- defaultNetworkUniversalTfProfile
- defaultNetworkL2Profile

The syntax to use non-default profiles when creating a network is given below. In the examples, network_name signifies the name of the network followed by a sub-string of the profile name:

- network_name:L2

- network_name: Ipv4Ef
- network_name: Ipv4Tf
- network_name: UniversalTf
- network_name: UniversalEf

Use defaultNetworkL2Profile

If this profile is chosen when a network is created in OpenStack, DCNM DHCP server will not assign an IP address for the VM associated with the network. Users are required to configure a static IP address for the VM. Additionally, the following command needs to be run on the OpenStack control node:

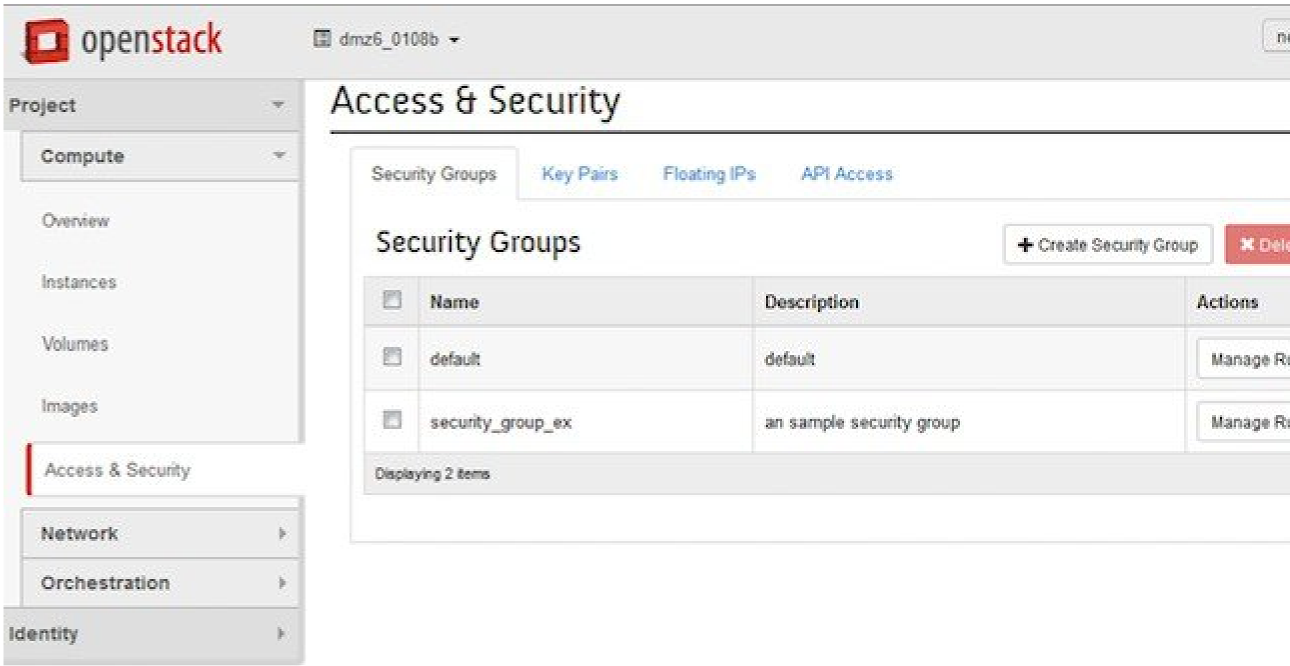
```
$fabric_enabler_cli
Cisco Nexus Fabric Command Line Interface
(Nexus-Fabric) set_static_ip --mac fa:16:3e:72:ab:dc --ip 136.10.0.16
```

The MAC address is the VM's vNIC and the IP address is the statically configured VM IP address. When a VM is removed from OpenStack, the above entry is automatically removed by the system.

Steps to Create a Security Group

You need to create and add a security group with appropriate rules before launching the VM. Create a security group and security rules that allow DHCP (from DCNM) and your data traffic to go through. After logging into Horizon as a user, click **Project** > **Compute** > **Access Security**. Use the **Create Security Group** tab to create a security group. After a security group is created, it appears in the **Security Groups** tab. A newly added group *security_group_ex* is displayed in the following sample screen shot.

Figure 78: Access and Security

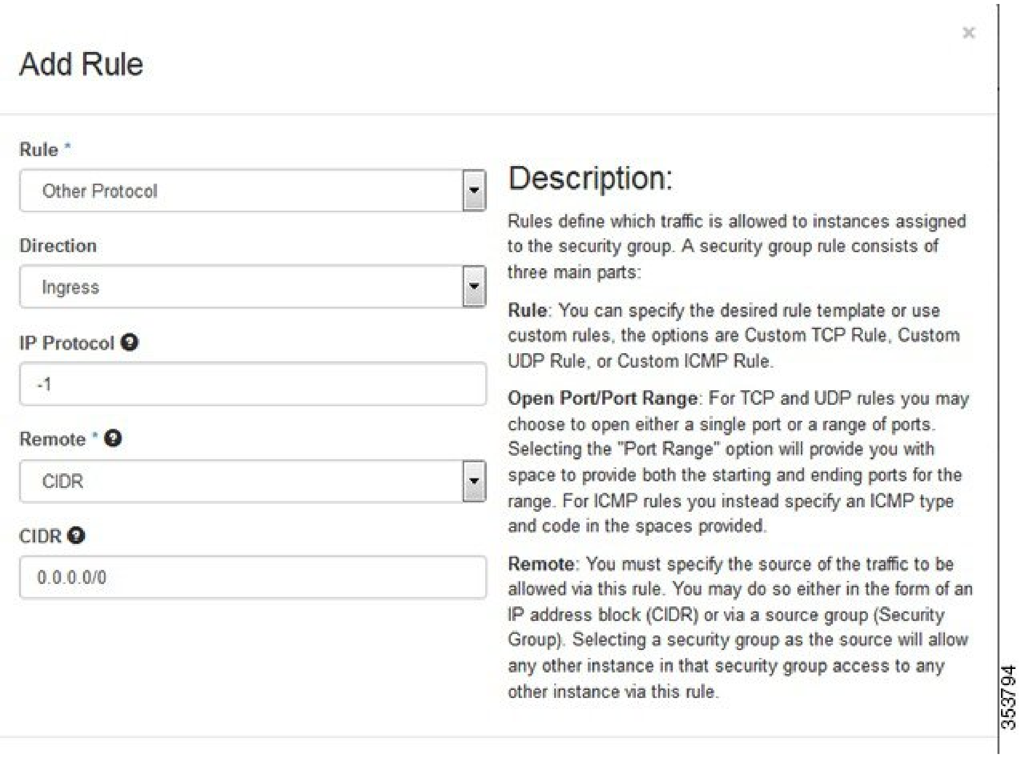


☐	Name	Description	Actions
☐	default	default	Manage R
☐	security_group_ex	an sample security group	Manage R

Displaying 2 items

Click **Manage Rules** for the security group you just created and add new rules. For example, if the following rule displayed in the *Add Rule* screen shot is added for the security group, it will allow all traffic.

Figure 79: Add Rule



Add Rule

Rule *

Other Protocol

Direction

Ingress

IP Protocol

-1

Remote *

CIDR

CIDR

0.0.0.0/0

Description:

Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Steps to Launch the VM

Follow these steps to launch the VM:

1. Click **Instances** and then click **Launch Instance**.
2. Click the **Image** drop-down menu and select the image.

   By default, the *CirrOS* image is selected.
3. Specify a name for the Instance.
4. Select the **Security** tab and choose the security group created (it is recommended to uncheck the default rule and select the one you specified).
5. Click the **Networking** tab and select the network from the **Available network** list.
6. Click **Launch**.

Limitations and Caveats

Known limitations and caveats are given below:

- Null Field Exception does not support IPv6.

- The VMs' internal vNIC interface state (*up* or *down*) is not detected by VDP. (Use VM creation or deletion as a potential workaround.)
- Servers with Converged Network Adapters (CNA) are not supported with OpenStack. The user needs to use a NIC.
- OpenStack DHCP is supported, while Cisco DCNM DHCP is not supported.

Technical Support

Following is the technical support model for using OpenStack:

- OpenStack is an on open source software platform, and is generally supported through its community using a best effort approach. If you use a third-party OpenStack installer and their support model, all generic OpenStack related support should come from the third-party support license.
- Support for the OpenStack Cisco Nexus Fabric Enabler part is provided from the Cisco Nexus Solution Team.

CHAPTER 15

Layer-4 to Layer-7 Services

Overview

Layer-4 through Layer-7 services support(s) end-to-end communication between a source and destination application. Layer 4-7 services are independent of the underlying network and provide services by enabling applications of load balancing, WAN acceleration and security appliances.

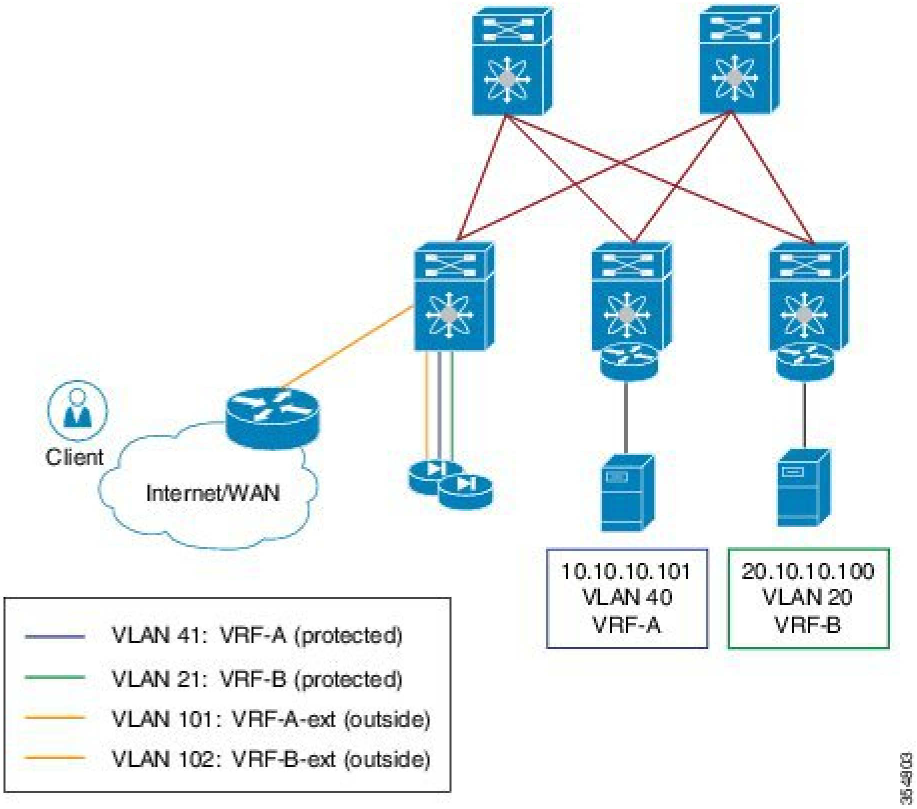
Cisco OpenStack Nexus Fabric Enabler (NFE) is a software that binds the generic OpenStack functionality with Nexus Fabric to provide the integrated and seamless cloud solution. The Enabler software consists of Nexus Server, Nexus Agent, and open source LLDP/VDP. For more information on Cisco Nexus Fabric, see Cisco Nexus Fabric OpenStack Enabler Install Guide.

Neutron is an OpenStack project to provide ‘networking as a service’ between interface devices (for example, virtualized Network Interface Card, used by a VM as its network interface) managed by other OpenStack services (for example, Nova). For more information on Neutron and related plugins, see Neutron.

Tenant Edge-Firewall

In a fabric, the firewall will allow or deny traffic based on the policies defined by the edge device. It will be used to control the north to south traffic, from a client to a datacenter server. The firewall requires of symmetry for ingress and egress traffic, whether it is an active/stand-by or clustering. A firewall is always connected to a router through a fabric to scale the resources.

Edge firewall connects a tenant (VRF or outside fabric) to another network in the north south traffic direction. VRF stitching is supported between the firewall and the server VRF with a transit subnet for routing. Servers use the anycast gateway (ToR) as the default gateway.


Client
Internet/WAN
10.10.10.101
VLAN 40
VRF-A
20.10.10.100
VLAN 20
VRF-B
VLAN 41: VRF-A (protected)
VLAN 21: VRF-B (protected)
VLAN 101: VRF-A-ext (outside)
VLAN 102: VRF-B-ext (outside)
354903

- Consider a client outside the fabric wants to access a server within tenant VLAN 40 in VRF-A or tenant VLAN 20 in VRF-B.
- Create a transit VLAN 41 in VRF-A for stitching and similarly VLAN 21 in VRF-B.
- Traffic from client reaches the router with the destination as VLAN 40 and the next hop is VLAN 101 (VRF-C).
- The Firewall intelligence routes the traffic to the connected Leaf on the protected Inside VLAN 41.
- The Service Leaf node (connected to the FW) routes the traffic over the fabric to the destination Leaf.
- Reverse traffic from Server in VLAN 40 uses the anycast-gateway IP on connected Leaf as next-hop. Default route injected by the FW is used to route traffic back to the Service Leaf node, which then routes it out to VLAN 101 towards the external router.

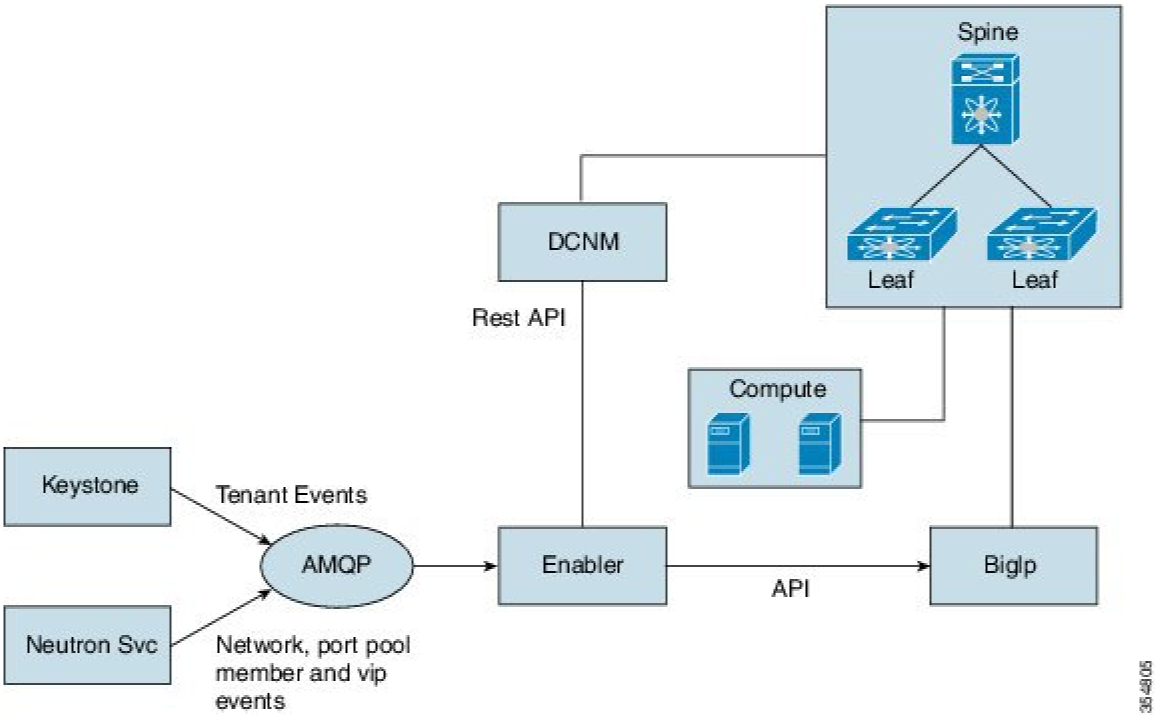
LBaaS

OpenStack supports LBaaS (Loadbalancer-as-a-Service). A virtual or physical device like 'f5 BIG-IPs' (a global load balancing solution) virtual and physical load balancer can be inserted into Cisco Programmable Fabric, allowing VMs acting as real servers behind the load balancer’s Virtual IPs (VIP), hosting various applications. By attaching the LBaaS to the Cisco Programmable Fabric network via its leaf switch is automatically done via auto-configuration and DCNM load balancer service VRF and network templates.

LBaaS Integration

OpenStack enabler does not use Layer-3 agent or neutron routers. All the Layesr-3 functions are provided by spine-leaf fabric clusters. Neutron provides services where the vendors can plug-in their own implementations of APIs and optionally bring up a service agent, which can communicate with neutron server via Remote Procedure Call (RPC) queue.

To integrate LBaaS to the fabric enabler, the service is installed with the fabric enabler installation.



Cisco enabler will listen to VIP, pool, member resource change events from neutron and program BigIP with the event information. As of NFE 2.0, LBaaS v1 is supported with BigIp virtual or physical box.

LBaaS Configuration

In the *neutron.conf* file, service provider must point to an object in enabler [service_providers]

service_provider=LOADBALANCER:F5:dfa.server.services.loadbalance.lb_enabler.LBEnabler:default.

Following is the pre-configuration required on the F5 loadbalancer.

- Configure the loadbalancer section within the *fabric_enabler.ini* file.
- Configure the IP address and gateway on the loadbalancer.
- Ensure loadbalancer is reachable via SSH and HTTPS.
- Create the same user on the loadbalancer, as configured in the *enabler_config.ini* file.
- Connect loadbalancer to the Nexus Fabric.

Following example shows the updated *fabric_enabler.ini* file.

```
lb_enabled = true
#lb_vrf_profile = vrf-common-universal-dynamic-LB-ES
```

```
lb_vrf_profile = vrf-common-evpn-dynamic-LB-ES
#lb_svc_net_profile=serviceNetworkUniversalDynamicRoutingLBProfile
lb_svc_net_profile=serviceNetworkDynamicRoutingLBEvpnProfile
#lb_svc_net = 199.199.1.0/24
#lb_svc_net_name_prefix=lbaasInternal
#comma separated (no space) list of big ip boxes
lb_mgmt_ip = 172.28.10.180
lb_user_name = admin
lb_user_password = cisco123
lb_f5_interface = N6k-73-75
[dcnm]
vlan_id_min = 670
vlan_id_max = 770
```

Note Ensure to uncheck ‘enable dhcp’ while creating VIP network.

LBaas Agent

To configure loadbalance service via OpenStack horizon dashboard or via neutron cli, you must have LBaaS driver plugins. You have two options to do this:

- Use OpenStack built in reference driver HAproxy and start its agent. The limitation is that as HAproxy is creating an internal interface with VIP address on br-int, you cannot use server subnet as VIP subnet because the traffic to VIP could be intercepted.
- Write a no-op plugin driver and there is no LBaaS agent.

BIG-IP Support

The services enabler should be able to support multiple Load balancer devices in the fabric. To support this, a pool of Load balancer IP Addresses can be configured into the enabler configuration file as follows.

```
[lbaas]
lb_mgmt_ip = 172.28.10.180, 172.28.10.181
lb_svc_net = 199.199.1.0/24, 188.188.1.0/24
```

For each project when a load balancer is allocated, a round robin algorithm can be used to pick one of the BIG-IP devices. Once selected, all the load balancers for that project are allocated to that BIG-IP device. Currently number of BIG-IP devices for a project will be limited to one. But multiple projects can use the same BIG-IP device. Fabric Enable is capable of restarting, to continue this support Project to BIG-IP module mapping will be stored in the SQL database via a new table.

When the fabric enabler service is started, the BIG-IP module of the enabler will establish connection with all the devices listed in the configuration file. Whenever a load balancer is created, enabler invokes the BIG-IP module with the following information.

```
[vlan, subnet IP, SubnetMask, Project Context ID, Project Context Name]
```

The BIG-IP module will pick one of the BIG-IPs and creates the required configuration. The Project context ID to BIG-IP device mapping is stored in the database in case the enabler restarts.

FWaaS

OpenStack supports FWaaS (Firewall-as-a-Service) that allows users to add perimeter firewall management to networking. FWaaS supports one firewall policy and logical firewall instance per project/tenant. With Cisco

Programmable Fabric, each compute node is attached to a leaf node, which hosts the physical default gateways for all the VMs on the compute node.

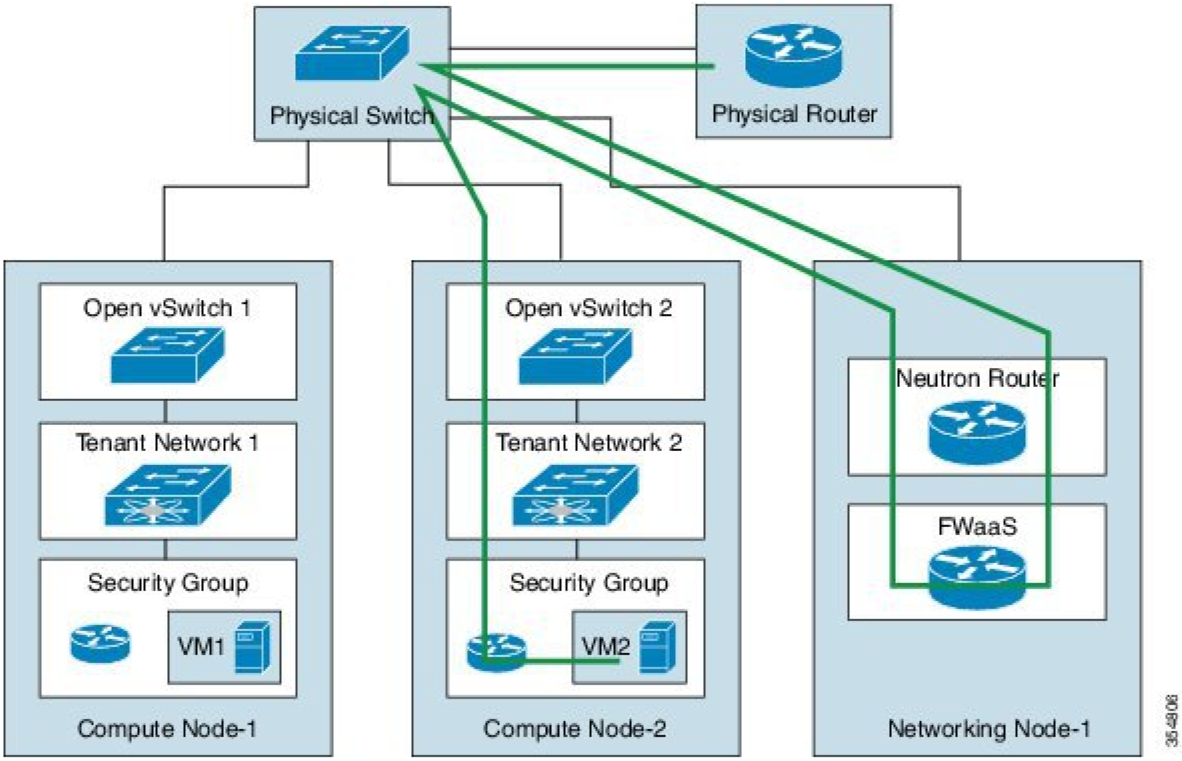
A VM based Firewall such as CSR1000v can be inserted into the Cisco Programmable Fabric architecture and used as FWaaS in OpenStack. FWaaS insertion into the fabric will be automatically triggered via OpenStack FWaaS functionality, including configuring the firewall rules and policy. The leaf switch will be configured using its auto-configuration feature and VRF and network templates from DCNM for both its internal VRF/network and external VRF/network for a given tenant.

While the initial effort is based on a Cisco Adaptive Security Appliance (ASA) firewall, the same architecture can allow the insertion of VM based firewall device into the fabric as well. Since the firewall is actually acting as a router, OSPF will be transparently programed by the enabler when FWaaS is provisioned from OpenStack.

Note Only physical ASA will be supported in first release. CSR1000v (virtual firewall) is not supported.

FWaaS Integration

OpenStack FWaaS adds perimeter firewall management to Networking. In OpenStack, routing is achieved through a router plugin. The router plugin uses the underlying Linux kernel routing implementation, to achieve routing. Every router created in OpenStack, has a Linux namespace created for it. If a VM in a subnet wants to communicate with another subnet, those subnets are added as interfaces to the router. In Juno release, every project is limited to one Firewall. The Firewall can have a policy, which can have multiple rules. The rules specify the classifier and actions. The firewall rules are applied to all the routers in the project and further to all the interfaces of a router.



OpenStack Firewall with Nexus Fabric

You can have firewall modules from different implementations with Nexus Fabric. When OpenStack is integrated with the Nexus Fabric, it is generally used as a tenant edge firewall. The following needs to be supported:

- Physical or a virtual firewall appliance plug-in.
- Configuration in OpenStack firewall must be translated to the appropriate firewall appliance's configuration.
- Nexus Fabric and the firewall appliance should have the appropriate configuration to support data path across the firewall appliance. This means:
    - Nexus Fabric should be able to redirect the traffic from a tenant, to the firewall module, if Tenant-Edge Firewall is configured for the tenant.
    - The firewall after performing the necessary operations, should have enough information to forward the frame to the fabric.
    - Similarly, the Nexus Fabric should forward the traffic from outside the tenant to the firewall module, which after performing the necessary operations should be able to forward the frame to the fabric, which can forward it back to the tenant.

The following section describes the packet flow for a tenant when a perimeter firewall is added through OpenStack in the presence of Physical ASA device acting as the firewall.

Firewall Manager

Firewall manager is responsible for acting on firewall related events from OpenStack such as policies, rules and Firewall creation/deletion and modification. The firewall and its associated rules and policies are decoded and stored per tenant. Based on the configuration, this module dynamically loads the Firewall Driver module. This module calls the Firewall Driver module when a firewall configuration gets added/deleted or modified.

For creating a Firewall, you need to create the rules, policy containing the rules and a Firewall containing the policy. You can create this in any order. For example, a Firewall can be created without any policy attached and later on, policies can be attached to the Firewall. Similarly, policies can be created without any rules attached at the beginning and then the policies can be updated with the rules. So, the Firewall module is responsible for caching the information and calling the modules to prepare the fabric or configure the Firewall device only after the complete firewall information is obtained.

Firewall Module

Firewall module is responsible for interacting with the actual device and doing the configuration. Depending on the appliance, the driver may use REST API or CLIs. Cisco Physical ASA and OpenStack Native firewall are supported.

Firewall module is responsible for the following:

- Set-up the fabric when a firewall is created.
- Clear the fabric configuration when the firewall is deleted.
- Provide APIs on information about the fabric for other modules including the driver to use.

Firewall Configuration

Following are the steps to setup ASA.

Step 1 Install Cisco ASA 5585-X. For more information, see Cisco ASA 5585-X Quick Start Guide.

The management interface can be set up by either using a console port or ADSM by connecting a PC to management interface. The ASA management IP address should be in the same management network as OpenStack Control and Compute nodes. You can either install a local ADSM client (mac) or run a web version.

Step 2 Ensure you run the recent version of Cisco ASA 5585-X.

Step 3 Install REST plugin.

Step 4 Acquire a Cisco ASA 5585-X license to support 100 security contexts.

Step 5 Enable Multiple Context Mode on ASA.

Step 6 Set up SSH access for Cisco ASA 5585-X. Ensure you can SSH to ASA. The management IP and credentials are required for OpenStack fabric enabler configuration (*enabler_cfg.ini*) file.

Step 7 Connect Cisco ASA 5585-X to leaf switch via two options:

- Connect one ASA physical port to a leaf switch or
- Connect two ASA physical ports to a pair of vPC-peer leaf switches (one to each). A port-channel shall be created on ASA for these two physical ports.

Example

Following example shows the configuration for the enabler software in the enabler_conf.ini file to configure the fabric and the device when a firewall is launched.

```
device = phy_asa
fw_mgmt_ip = '3.3.3.5'
#fw_username = [admin]
#fw_password = [cisco123]
#fw_interface_in = [Gi0/0]
#fw_interface_out = [Gi0/1]
#fw_auto_serv_nwk_create = True
# Currently only the MAX scheduling is supported
#sched_policy = 'max_sched'
#mob_domain_name = 'md0'
#fw_service_host_profile = 'serviceNetworkUniversalDynamicRoutingESProfile'
#fw_service_host_fwd_mode = 'proxy-gateway'
#fw_service_part_vrf_profile = 'vrf-common-universal-external-dynamic-ES'
#fw_service_ext_profile = 'externalNetworkUniversalDynamicRoutingESProfile'
#fw_service_ext_fwd_mode = 'anycast-gateway'
# It assumes a 24 subnet. In the example below, the first service IN network
# will have 100.100.2.x, and subsequent ones with have 100.100.3.x and so on.
#fw_service_in_ip_start = 100.100.2.0
#fw_service_in_ip_end = 100.100.20.0
#fw_service_out_ip_start = 200.200.2.0
#fw_service_out_ip_end = 200.200.20.0
#fw_service_dummy_ip_subnet = '9.9.9.0/24'
```

```
[firewall]
# Firewall Default Parameters
```

ASA Firewall driver

The firewall physical driver is the software layer that handles firewall events from firewall manager and program firewalls accordingly. Each type of firewalls, both physical and virtual, requires its own physical driver.

Firewall physical driver uses the REST API to configure the Cisco ASA 5585-X.

Event Handling

The ASA physical driver handles the following OpenStack events from firewall manager:

- Firewall creation
- Firewall deletion
- Add firewall
- Modify firewall
- Delete firewall

REST API support

The Cisco ASA 5585-X REST API supports only a set of basic features, such as access rules and interfaces, but it provides a special API to send any CLI command to ASA.

This special API can take single or multi-line CLI commands and will present the output of the CLI as the API response:

```
Post URL:
/api/cli
Post Request Payload/Content:
{
"commands": ["command-1", "command-2",..., "command-n"]
}
Response:
{
"response": ["command-1 response", "command-2 response",..., "command-n response"]
}
```

The ASA physical driver implements a wrapper function (asa_rest_send) around this API to send commands to ASA 5585.

ASA Configuration

The ASA physical driver settings (IP of management interface and login credentials) are saved in “fw_constants.py”. The *enabler_conf.ini* file must have the following configuration enabled to support native firewall.

```
fw_service_host_profile = serviceNetworkUniversalESChainLBESProfile
fw_service_part_vrf_profile = vrf-common-universal-external- dynamic-ES
fw_service_ext_profile = externalNetworkUniversalDynamicRoutingESProfilew
```

- fw_service_host_profile is the profile for the IN service network created by firewall module in Cisco DCNM.

- fw_service_part_vrf_profile is the profile of the OUT service partition created by firewall module in Cisco DCNM.
- fw_service_ext_profile is the profile for the OUT service network created by firewall module in Cisco DCNM.

Limitation

- Cisco ASA 5585-X can only support up to 100 security contexts with a proper license, the maximum number of tenants that an OpenStack deployment can support with one Cisco ASA 5585-X is up to 100.
- 125 VRF with 1 OSPF is the max that has been tested on Cisco Nexus 6000. Cisco ASA 5585 can support up to 2 OSPF instances. Each ASA security context can have their own OSPF configuration.

OpenStack Native Firewall driver

The driver handles the firewall events from firewall manager and programs the firewall accordingly.

OpenStack firewall support is achieved using Linux IP table rules. For Layer-3 forwarding, OpenStack relies on Linux kernel Layer-3 stack and uses namespace to achieve tenant separation. Natively, when a firewall is created in OpenStack, the rules of the firewall are applied to all the interfaces of the router. Creation of a router and attaching the networks to a router is a separate operation. Integrating OpenStack native firewall to a standalone Cisco fabric is achieved as follows:

- When a Firewall is created in OpenStack, the driver automatically creates a router and attaches the IN or OUT interfaces to the router.
- Driver ensures auto-configuration for the two interfaces is done in the fabric to ensure traffic is forwarded to the interfaces.
- Unlike the Physical ASA, the router namespace does not run any routing protocol like OSPF, hence the driver ensures that static routes are populated in the fabric through Cisco DCNM.

Event handling

The native OpenStack firewall driver handles the following OpenStack events from firewall manager:

- Firewall creation
- Firewall deletion
- Modify firewall

Firewall Creation

Creates a firewall instance in OpenStack, the driver performs the following tasks.

1. Creates an OpenStack router using OpenStack python APIs. Attach the IN and OUT subnets to the router.
2. Programs Cisco DCNM to update profile's static IP address on OUT partition. Since there is no OSPF running between the router namespace and the switch, the routes should be statically configured.
3. Programs the default gateway in router namespace to point to the OUT interface.
4. Both the IN and OUT subnet interfaces of the router belong to different VRF or Segment. For the switch to forward traffic to the router namespace, auto-config should happen just like a regular VM and these router interfaces should be seen as a VSI in the switch.

Firewall Deletion

Deletes the firewall instance OpenStack, the driver performs the following tasks.

1. Removes the OpenStack router.
2. Sends a VM VNIC delete message to the enabler event queue with the right segmentation ID for both IN and OUT interfaces of the router namespace.

Modify Firewall

The driver does not act on this event since OpenStack internally programs the router namespace.

Deployment

Enabler minimal configuration

The *enabler_conf.ini* file must have the following configuration enabled to support native firewall.

```
fw_service_host_profile = 'serviceNetworkUniversalESChainLBESProfile '
fw_service_part_vrf_profile = 'vrf-common-universal-external-static'
fw_service_ext_profile = 'externalNetworkUniversalTfStaticRoutingESProfile'
```

- fw_service_host_profile is the profile for the IN service network created by firewall module in Cisco DCNM.
- fw_service_part_vrf_profile is the profile of the OUT service partition created by firewall module in Cisco DCNM.
- fw_service_ext_profile is the profile for the OUT service network created by firewall module in Cisco DCNM.

Enabler Restart

When enabler restarts, it retrieves configuration or state information from DB (MySQL) to initialize. Since only the create or delete events are handled by the driver for native firewall, the restart functionality is minimal.

Limitation

- For every router creation, OpenStack creates a namespace.
- Firewall configurations are done inside the namespace. The router namespace exists only in the network node of OpenStack.
- OpenStack also has HA for network nodes. This native firewall solution is not tested with HA or multiple network nodes. Since, all the OUT traffic needs to flow through a network node, there can be scalability issues when there is heavy traffic that needs firewall policies to be applied since the firewall is not distributed (if firewall ran as a VM instead of a namespace) and runs in the Linux kernel which can be slower than a custom build ASIC.