



Inspur

CN12700 Series

INOS VXLAN Configuration Guide

(Release 8.x)



Inspur-Cisco Networking Technology Co.,Ltd. provides customers with comprehensive technical support and services.
For any assistance, please contact our local office or company headquarters.

Website: <http://www.inspur.com/>

Technical Support Tel: 400-691-1766

Technical Support Email: inspur_network@inspur.com

Technical Document Support Email: inspur_network@inspur.com

Address: 1036 Langchao Road, Lixia District, Jinan City, Shandong Province

Postal code: 250101

Notice

Copyright © 2020

Inspur Group.

All rights reserved.

No part of this publication may be excerpted, reproduced, translated or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in Writing from **Inspur-Cisco Networking Technology Co.,Ltd.**

inspur 浪潮

is the trademark of **Inspur-Cisco Networking Technology Co.,Ltd.**

All other trademarks and trade names mentioned in this document are the property of their respective holders.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute the warranty of any kind, express or implied

Preface

Objectives

This guide describes main functions of the CN12700 Series. To have a quick grasp of the CN12700 Series, please read this manual carefully.

Versions





The following table lists the product versions related to this document.

Product name	Version
CN12700 Series	

Conventions

Symbol conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
 Warning	Indicates a hazard with a medium or low level of risk which, if not avoided, could result in minor or moderate injury.
 Caution	Indicates a potentially hazardous situation that, if not avoided, could cause equipment damage, data loss, and performance degradation, or unexpected results.
 Note	Provides additional information to emphasize or supplement important points of the main text.
 Tip	Indicates a tip that may help you solve a problem or save time.

General conventions

Convention	Description
Boldface	Names of files, directories, folders, and users are in boldface . For example, log in as user root .
Italic	Book titles are in <i>italics</i> .
Lucida Console	Terminal display is in <code>Lucida Console</code> .

Command conventions

Convention	Description
Boldface	The keywords of a command line are in boldface .
Italic	Command arguments are in <i>italics</i> .
[]	Items (keywords or arguments) in square brackets [] are optional.
{ x y ... }	Alternative items are grouped in braces and separated by vertical bars. One is selected.
[x y ...]	Optional alternative items are grouped in square brackets and separated by vertical bars. One or none is selected.
{ x y ... } *	Alternative items are grouped in braces and separated by vertical bars. A minimum of one or a maximum of all can be selected.
[x y ...] *	The parameter before the & sign can be repeated 1 to n times.

GUI conventions

Convention	Description
Boldface	Buttons, menus, parameters, tabs, windows, and dialog titles are in boldface . For example, click OK .
>	Multi-level menus are in boldface and separated by the ">" signs. For example, choose File > Create > Folder .

Keyboard operation

Format	Description
Key	Press the key. For example, press Enter and press Tab .

Format	Description
Key 1+Key 2	Press the keys concurrently. For example, pressing Ctrl+C means the two keys should be pressed concurrently.
Key 1, Key 2	Press the keys in turn. For example, pressing Alt, A means the two keys should be pressed in turn.

Mouse operation

Action	Description
Click	Select and release the primary mouse button without moving the pointer.
Double-click	Press the primary mouse button twice continuously and quickly without moving the pointer.
Drag	Press and hold the primary mouse button and move the pointer to a certain position.

Change history

Updates between document versions are cumulative. Therefore, the latest document version contains all updates made to previous versions.

Issue 01 (2020-02-24)

Initial commercial release

Contents

Contents	v
Figures	viii
Tables	ix
CHAPTER 1 Overview	1
1.1 VXLAN Overview.....	1
1.2 VXLAN Flood and Learn.....	1
1.3 VXLAN MAC Distribution.....	2
1.4 VXLAN Tunnel Endpoint.....	2
1.5 Virtual Network Identifier (VNI).....	2
1.6 VXLAN BGP EVPN Control Plane.....	3
CHAPTER 2 Configuring VXLAN Flood and Learn	4
2.1 Information About VXLAN.....	4
2.2 Guidelines and Limitations for VXLAN.....	6
2.3 Configuring VXLAN.....	11
2.4 Verifying the VXLAN Configuration.....	17
2.5 Configuration Examples.....	17
2.6 Feature History for VXLAN Flood and Learn.....	27
CHAPTER 3 Configuring VXLAN BGP EVPN	28
3.1 Information About VXLAN BGP EVPN.....	28
3.2 Configuring VXLAN BGP EVPN.....	39
3.3 Feature History for VXLAN BGP EVPN.....	46
CHAPTER 4 Configuring ACI WAN Interconnect	47
4.1 VXLAN EVPN - MPLS L3VPN for ACI Fabric.....	47
CHAPTER 5 Campus Fabric	63
5.1 Campus Fabric.....	63
5.1.1 Overview of Campus Fabric.....	63
5.2 Feature History for Campus Fabric.....	67
CHAPTER 6 Campus Fabric Interconnect - MPLS L3VPN	68
6.1 Prerequisites of Campus Fabric Interconnect—MPLS L3VPN.....	68

6.2 Information About Campus Fabric Interconnect—MPLS L3VPN	68
6.3 How to Configure Campus Fabric Interconnect—MPLS L3VPN.....	69
6.4 Verifying Campus Fabric Interconnect—MPLS L3VPN.....	71
6.5 Feature History for Campus Fabric Interconnect—MPLS L3VPN.....	72
CHAPTER 7 LISP Support for Disjointed RLOC Domains.....	73
7.1 LISP Support for Disjointed RLOC Domains.....	73
CHAPTER 8 PBR support for the VXLAN BGP EVPN fabric.....	81
8.1 Prerequisites for PBR Support for the VXLAN BGP EVPN Fabric.....	81
8.2 Guidelines and Limitations for PBR Support for the VXLAN BGP EVPN Fabric.....	81
8.3 Information About PBR Support for the VXLAN BGP EVPN Fabric.....	81
8.4 How to Configure PBR Support for the VXLAN BGP EVPN Fabric.....	83
8.5 Verifying PBR Support for VXLAN BGP EVPN Fabric.....	83
8.6 Feature History for PBR Support for the VXLAN BGP EVPN Fabric.....	84
CHAPTER 9 VXLAN BGP EVPN and OTV Interoperation.....	85
9.1 Prerequisites for VXLAN BGP EVPN and OTV Interoperation.....	85
9.2 Guidelines and Limitations for VXLAN BGP EVPN and OTV Interoperation.....	85
9.3 Information About VXLAN BGP EVPN and OTV Interoperation.....	86
9.4 How to Configure VXLAN BGP EVPN and OTV Interoperation, and OTV with BDI.....	90
9.5 Verifying VXLAN BGP EVPN and OTV Interoperation, and OTV with BDI.....	93
9.6 Feature History for VXLAN BGP EVPN and OTV Interoperation.....	97
CHAPTER 10 Proportional Multipath for VNF.....	99
10.1 Information About Proportional Multipath for VNF.....	99
10.2 Licensing Requirements for Proportional Multipath for VNF.....	100
10.3 Guidelines and Limitations for Proportional Multipath for VNF.....	101
10.4 Default Setting for Proportional Multipath for VNF.....	101
10.5 Configuring Proportional Multipath for VNF.....	102
10.6 Verifying Proportional Multipath for VNF.....	108
10.7 Configuration Examples for Proportional Multipath for VNF.....	120
10.8 Additional References for Proportional Multipath for VNF.....	122
10.9 Feature History for Proportional Multipath for VNF.....	122
CHAPTER 11 Centralized VRF Route-Leaking for VXLAN BGP EVPN Fabrics.....	123
11.1 Configuring Route Leaking.....	124

11.2 Configuring Routes Imported from a VPN to Leak into the Default VRF.....	139
11.3 Configuring Routes Leaked from the Default-VRF to Export to a VPN.....	139
11.4 Configuring Routes Imported from a VPN to Export to a VRF.....	140
11.5 Configuring Routes Imported from a VRF to Export to a VPN.....	140
11.6 Configuration Examples.....	141
11.7 Displaying Centralized Route Leaking Information.....	144
11.8 Displaying Centralized Route Leaking Information.....	147
11.9 Feature History for Centralized VRF Route Leaking.....	149

Figures

<i>Figure 1 : Layer 2 VXLAN Topology.....</i>	<i>5</i>
<i>Figure 2 : Layer 2 Gateway Packet Header.....</i>	<i>5</i>
<i>Figure 3 : Layer 3 Gateway Packet Header.....</i>	<i>6</i>
<i>Figure 4 : VXLAN Overlay.....</i>	<i>29</i>
<i>Figure 5 : VXLAN Packet Format.....</i>	<i>30</i>
<i>Figure 6 : Inter Tenant Traffic Flow Using VRF VNI.....</i>	<i>31</i>
<i>Figure 7 : End Host IP + MAC Address Distribution in a VXLAN Setup.....</i>	<i>31</i>
<i>Figure 8 : New End Host Attaches to a VTEP.....</i>	<i>33</i>
<i>Figure 9 : Distributed Gateway.....</i>	<i>34</i>
<i>Figure 10 : Packet Forwarding (Bridge).....</i>	<i>34</i>
<i>Figure 11: Packet Forwarding (Route).....</i>	<i>35</i>
<i>Figure 12 : ARP Suppression.....</i>	<i>36</i>
<i>Figure 13 : End Host Detection.....</i>	<i>37</i>
<i>Figure 14 : Host A - Address Distribution Parameters.....</i>	<i>37</i>
<i>Figure 15 : Host move from V1 to V3.....</i>	<i>37</i>
<i>Figure 16 : Host A – Updated Parameters.....</i>	<i>38</i>
<i>Figure 17 : ACI Fabric & L3VPN Hand-off.....</i>	<i>48</i>
<i>Figure 18 : Single POD with Multiple DCI Gateways.....</i>	<i>49</i>
<i>Figure 19 : Multi-POD with Shared DCI Gateway.....</i>	<i>49</i>
<i>Figure 20 : Multi-POD with Separate DCI Gateways.....</i>	<i>50</i>
<i>Figure 21 : DCI Auto-Configuration].....</i>	<i>51</i>
<i>Figure 22: Campus Fabric Architecture.....</i>	<i>63</i>
<i>Figure 23 : Overall Topology of Campus Fabric.....</i>	<i>64</i>
<i>Figure 24 : Sample topology—LISP Support for Disjointed RLOC Domains.....</i>	<i>74</i>
<i>Figure 25: Sample topology—PBR support for the VXLAN BGP EVPN fabric.....</i>	<i>82</i>
<i>Figure 26 : Sample topology 1 - VXLAN BGP EVPN and OTV interoperation.....</i>	<i>86</i>
<i>Figure 27 : Sample topology 2 - VXLAN BGP EVPN and OTV interoperation.....</i>	<i>86</i>
<i>Figure 28 : Centralized VRF Route-Leaking - Specific Prefixes with Custom VRF.....</i>	<i>125</i>
<i>Figure 29 : Centralized VRF Route-Leaking - Shared Internet with Custom VRF.....</i>	<i>128</i>
<i>Figure 30 : Centralized VRF Route-Leaking - Shared Internet with VRF Default.....</i>	<i>134</i>

Tables

<i>Table 1 : Feature History for VXLAN Flood and Learn.....</i>	<i>27</i>
<i>Table 2 : Feature History for VXLAN BGP EVPN.....</i>	<i>46</i>
<i>Table 3: Feature History for ACI WAN Interconnect.....</i>	<i>47</i>
<i>Table 4 : Feature History for Campus Fabric.....</i>	<i>67</i>
<i>Table 5 : Feature History for Campus Fabric Interconnect—MPLS L3VPN.....</i>	<i>72</i>
<i>Table 6 : Feature History for LISP Support for Disjointed RLOC Domains.....</i>	<i>80</i>
<i>Table 7 : Feature History for PBR Support for the VXLAN BGP EVPN fabric.....</i>	<i>84</i>
<i>Table 8 : Feature History for VXLAN BGP EVPN and OTV interoperation.....</i>	<i>98</i>
<i>Table 9 : Feature History for IP TCP MSS.....</i>	<i>149</i>

CHAPTER 1 Overview

This chapter contains the following sections:

- VXLAN Overview.
- VXLAN Flood and Learn.
- VXLAN MAC Distribution.
- VXLAN Tunnel Endpoint.
- Virtual Network Identifier (VNI).
- VXLAN BGP EVPN Control Plane .

1.1 VXLAN Overview

VXLAN is a MAC in IP/UDP(MAC-in-UDP) encapsulation technique with a 24-bit segment identifier in the form of a VXLAN ID. The larger VXLAN ID allows LAN segments to scale to 16 million in a cloud network. In addition, the IP/UDP encapsulation allows each LAN segment to be extended across existing Layer 3 networks making use of Layer 3 equal-cost multipath (ECMP).

Inspur CN12700 switches are designed for hardware-based VXLAN function. It provides Layer 2 connectivity extension across the Layer 3 boundary and integrates between VXLAN and non-VXLAN infrastructures.

This can enable virtualized and multi tenant data center designs over a shared common physical infrastructure.

VXLAN provides a way to extend Layer 2 networks across Layer 3 infrastructure using MAC-in-UDP encapsulation and tunneling. VXLAN enables flexible workload placements using the Layer 2 extension. It can also be an approach to building a multi tenant data center by decoupling tenant Layer 2 segments from the shared transport network.

When deployed as a VXLAN gateway, Inspur CN12700 switches can connect VXLAN and classic VLAN segments to create a common forwarding domain so that tenant devices can reside in both environments.

VXLAN has the following benefits:

- Flexible placement of multi tenant segments throughout the data center.

It provides a way to extend Layer 2 segments over the underlying shared network infrastructure so that tenant workloads can be placed across physical pods in the data center.

- Higher scalability to address more Layer 2 segments.

VXLAN uses a 24-bit segment ID, the VXLAN network identifier (VNID). This allows a maximum of 16 million VXLAN segments to coexist in the same administrative domain. (In comparison, traditional VLANs use a 12-bit segment ID that can support a maximum of 4096 VLANs.)

- Utilization of available network paths in the underlying infrastructure.

VXLAN packets are transferred through the underlying network based on its Layer 3 header. It uses equal-cost multipath (ECMP) routing and link aggregation protocols to use all available paths.

There are two unicast modes in which VXLAN can run. They are Flood and Learn mode and MAC Distribution mode.

1.2 VXLAN Flood and Learn

VXLAN is MAC in IP/UDP encapsulation technique with a 24-bit segment identifier in the form of a VXLAN ID. The larger VXLAN ID allows LAN segments to scale to 16 million in a cloud network. In addition, the IP/UDP encapsulation allows each LAN segment to be extended across existing Layer 3 network. Traditionally with virtual VTEPS the only endpoints that can connect into VXLANs are physical or virtual connections. Physical servers cannot be in the VXLAN network. Routers or services that have traditional VLAN interfaces cannot be used by VXLAN-based networks.

The VXLAN flood and learn gateway feature provides solution to this problem. VXLAN flood and learn gateway enables the following:

- Host learning on VTEPs based on flood and learn behaviour
- VTEPs join underlay IP multicast groups based on VNI ‘membership’
- If VNI exists behind VTEP, the packet flow joins the corresponding IP multicast group in underlay
- ARP (and other broadcast / unknown unicast / multicast traffic) in a given VNI flooded to all interested VTEPs

VTEPs

- Gateway functions centralised in VXLAN flood and learn
- Inspur CN12700 vPC pair with L2 + L3 VXLAN gateway capabilities
- vPC provides MAC state synchronization and active-active HSRP forwarding
- Redundant VTEPs share Anycast VTEP IP address in underlay
- VXLAN bridging occurs directly between VTEPs

1.3 VXLAN MAC Distribution

In VXLAN MAC Distribution mode, head-end replication is used to deliver broadcast and multicast frames to the entire network.

1.4 VXLAN Tunnel Endpoint

VXLAN uses VXLAN tunnel endpoint (VTEP) devices to map tenants’ end devices to VXLAN segments and to perform VXLAN encapsulation and de-encapsulation. Each VTEP function has two interfaces: One is a switch interface on the local LAN segment to support local endpoint communication through bridging, and the other is an IP interface to the transport IP network.

The IP interface has a unique IP address that identifies the VTEP device on the transport IP network known as the infrastructure VLAN. The VTEP device uses this IP address to encapsulate Ethernet frames and transmits the encapsulated packets to the transport network through the IP interface. A VTEP device also discovers the remote VTEPs for its VXLAN segments and learns remote MAC Address-to-VTEP mappings through its IP interface.

The VXLAN segments are independent of the underlying network topology; conversely, the underlying IP network between VTEPs is independent of the VXLAN overlay. It routes the encapsulated packets based on the outer IP address header, which has the initiating VTEP as the source IP address and the terminating VTEP as the destination IP address.

1.5 Virtual Network Identifier (VNI)

In RFC 4364 L3VPNs, a 20-bit MPLS label that is assigned to a VPN route determines the forwarding behavior in the data plane for traffic following that route. These labels also serve to distinguish the packets of one VPN from another.

On the other hand, the various IP overlay encapsulations support a virtual network identifier (VNI) as part of their encapsulation format.

A VNI is a value that at a minimum can identify a specific virtual network in the data plane. It is typically a 24-bit value which can support up to 16 million individual network segments.

There are two useful requirements regarding the scope of these VNIs.

- Network-wide scoped VNIs

Depending on the provisioning mechanism used within a network domain such as a data center, the VNI may have a network scope, where the same value is used to identify the specific Layer-3 virtual network across all network edge devices where this virtual network is instantiated. This network scope is useful in environments such as within

the data center where networks can be automatically provisioned by central orchestration systems.

Having a uniform VNI per VPN is a simple approach, while also easing network operations (i.e. troubleshooting). It also means simplifies requirements on network edge devices, both physical and virtual devices. A critical requirement for this type of approach is to have a very large amount of network identifier values given the network-wide scope.

- Locally assigned VNIs

In an alternative approach supported as per RFC 4364, the identifier has local significance to the network edge device that advertises the route. In this case, the virtual network scale impact is determined on a per node basis, versus a network basis.

When it is locally scoped, and uses the same existing semantics of a MPLS VPN label, the same forwarding behaviors as specified in RFC 4364 can be employed. It thus allows a seamless stitching together of a VPN that spans both an IP based network overlay and a MPLS VPN.

This situation can occur for instance at the data center edge where the overlay network feeds into an MPLS VPN. In this case, the identifier may be dynamically allocated by the advertising device.

It is important to support both cases, and in doing so, ensure that the scope of the identifier be clear and the values not conflict with each other.

It should be noted that deployment scenarios for these virtual network overlays are not constrained to the examples used above to categorize the options. For example, a virtual network overlay may extend across multiple data centers.

- Global unicast table

The overlay encapsulation can also be used to support forwarding for routes in the global or default routing table. A VNI value can be allocated for the purpose as per the options mentioned above.

1.6 VXLAN BGP EVPN Control Plane

The EVPN overlay specifies adaptations to the BGP MPLS-based EVPN solution to enable it to be applied as a network virtualization overlay with VXLAN encapsulation where:

- PE node role described in BGP MPLS EVPN is equivalent to VTEP/network virtualization edge (NVE) device.
- VTEP information is distributed via BGP.
- VTEPs use control plane learning/distribution via BGP for remote MAC addresses instead of data plane learning.
- Broadcast, unknown unicast and multicast (BUM) data traffic is sent using a shared multicast tree.
- BGP route reflector (RR) is used to reduce the full mesh of BGP sessions among VTEPs to a single BGP session between a VTEP and the RR.
- Route filtering and constrained route distribution are used to ensure that the control plane traffic for a given overlay is only distributed to the VTEPs that are in that overlay instance.
- Host (MAC) mobility mechanism to ensure that all the VTEPs in the overlay instance know the specific VTEP associated with the MAC.
- Virtual network identifiers (VNIs) are globally unique within the overlay.

The EVPN overlay solution for VXLAN can also be adapted to enable it to be applied as a network virtualization overlay with VXLAN for Layer 3 traffic segmentation. The adaptations for Layer 3 VXLAN are similar to L2 VXLAN except the following:

- VTEPs use control plane learning/distribution via BGP of IP addresses (instead of MAC addresses)
- The virtual routing and forwarding instance is mapped to the VNI
- The inner destination MAC address in the VXLAN header does not belong to the host but to the receiving VTEP that does the routing of the VXLAN payload. This MAC address is distributed via BGP attribute along with EVPN routes.

CHAPTER 2 Configuring VXLAN Flood and Learn

This chapter contains the following sections:

- Information About VXLAN.
- Configuring VXLAN.
- Verifying the VXLAN Configuration.
- Configuration Examples.
- Feature History for VXLAN Flood and Learn.

2.1 Information About VXLAN

2.1.1 VXLAN with vPC Overview

The vSwitch can be dually connected to the vPC via a hypervisor VLAN. The VTEP on L1 and L2 is identified by the same IP address. vPC is active/active for East-West traffic (from the physical server to the VMs behind vSwitch). It is active/standby with the elected forwarder for packets from North-South traffic.

Active/Active scheme:

- Both switches perform encapsulation and decapsulation between the VXLAN tunnel (on the access side) and the physical servers. To prevent duplicate copies, we rely on the VSL bit.
 - If a packet comes in on hypervisor VLAN on a switch it is bridged to other peers via the peerlink. The decapsulated copy is prevented from going over the peer link (using reserved ftag CBL scheme). Both switches bridge locally on the hypervisor VLAN and decapsulate and bridge in the tenant VNI (the VSL bit prevents a duplicate copy from being sent to the vPC legs).
 - If a packet comes into the tenant VNI from the physical server, it is bridged to the vPC peer. The VXLAN tunnel encapsulated copy is blocked from going over the peerlink using LTL+1 logic.

Active/Standby scheme:

- Since hardware does not have a mechanism to prevent both switches from sending and receiving packets to and from the North, one of the vPC peers is selected as the forwarder by PIM.

2.1.2 VXLAN Layer 2 Gateway

The VXLAN Layer 2 gateway bridges traffic between physical servers and VM's behind vSwitches that are in the same VNI.

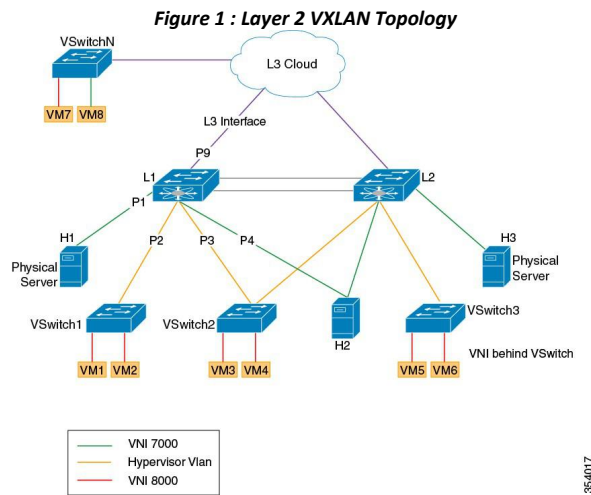
- Connectivity of vSwitches to Inspur CN12700 is via a Layer 2 port through a VLAN which is called a hypervisor VLAN. One of the requirements for a VXLAN gateway is that the hypervisor VLAN should be Layer 3 enabled (SVI configured) and be a member of the core VRF.
 - Traffic from the physical server is mapped to segment (VNI) using VSI configuration.
 - Traffic from VMs behind vSwitches are encapsulated in VXLAN format with VNI information from the server to which it belongs. The VNI identifies the bridge-domain that both the physical server and the virtual servers are a part.
 - For packets coming from vSwitches, the Layer 2 VXLAN gateway strips the VXLAN header and identifies the bridge-domain before bridging the packet to the physical server. Similarly, when physical servers talk to VM's behind vSwitches, the VXLAN header is appended with appropriate VNI information before sending it to the vSwitches.
 - VXLAN uses the control multicast group for broadcast, unknown unicast and multicast (BUM) traffic. When the control multicast group is configured on the vSwitch, it sends IGMP reports to the Inspur CN12700 switch on the hypervisor VLAN . This results in Layer 2 multicast state creation for the control multicast group on the hypervisor

VLAN. Since the hypervisor VLAN is Layer 3 enabled on the core VRF, it triggers a PIM join and Layer 3 multicast state creation. Thus, BUM traffic is bridged to locally connected vSwitches via Layer 2 multicast bridging and to remote vSwitches behind Layer 3 cloud via Layer 3 multicast routing.

2.1.3 VXLAN Layer 3 Gateway

Layer 3 VXLAN gateway enables routing between different VNIs. The Inspur CN12700 can be placed as a pure Layer 3 routing box, which does inter VNI routing or it can be placed along with Layer 2 VXLAN gateway functionality. To enable Layer 3 VXLAN functionality, BDI has to be configured on the tenant VNI and the tenant VRF has to be different from the core VRF.

All VMs and physical servers and VNIs belonging to the same tenants can communicate. Any packet that needs to be routed across VNIs needs to be sent to the Layer 3 gateway switch, by setting the outer IP to the Layer 3 gateway IP, and the inner DMAC to be the Layer 3 gateway MAC.



The implications of the limited Layer 3 gateway functionality are the following:

- Since the Layer 3 gateway is centralized, there is no need to run control protocols (to advertise the host reachability information). When the Layer 3 gateway receives the packet, it looks at the Layer 3 header information to route the packet (to the destination subnet/VNI), but the actual remote switch to which the packet needs to be forwarded is resolved at the Layer 2 level, which is done via data plane learning.

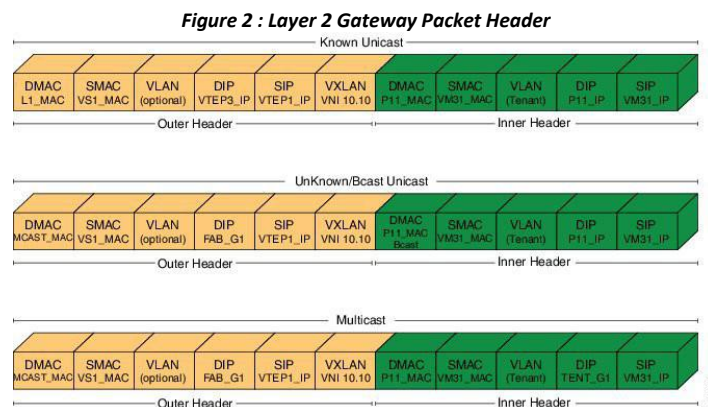


Figure 3 : Layer 3 Gateway Packet Header



For example, in the above diagram the packets from VM31 (1.1.1.x) to P12 (2.2.2.x) are resolved at the Layer 3 level to go to SVI_VNI_20.20. After the routing, the destination VTEP, that is responsible for the destination host, is identified at the Layer 2 level. There are two cases to consider here:

1. If the switch that runs the Layer 3 gateway functionality does not run Layer 2 gateway functionality, that is, participates in the data plane learning, it has to flood the post-routed packet to all Layer 2 gateways through the multicast tree.
2. If the switch runs Layer 3 gateway and Layer 2 gateway functionality, the switch can resolve the destination VTEP at the Layer 2 level, and can forward the packet to the correct VTEP by itself.

Since Layer 2 gateway and Layer 3 gateway functionalities are tightly integrated from the configuration perspective (Layer 3 gateway is achieved by configuring the BDIs for bridge-domains corresponding to the VNIs), case (1) will not be applicable in the Inspur CN12700, and only case (2) is supported.

Since there is a centralized Layer 3 gateway, any multicast packet that needs to be routed across multiple VNIs, will have to be replicated multiple times (one for each VNI).

In the above example, if P12 is interested in traffic initiated by VM31, the Layer 3 gateway will have to send two copies of the packet (one for VNI 20.20, and, one for VNI 30.30).

Also, note that routing between VNI and Layer 3 interfaces can be supported only for those interfaces that are local to the Layer 3 gateway, assuming those local Layer 3 interfaces are configured in the same tenant VRF as the VNI. One possible solution would be to have a separate Layer 3 connection between those leaf switches and spine (Layer 3 gateway) switch, configure them all in tenant VRF, run OSPF (or IS-IS) for that tenant VRF, and run PIM to draw multicast traffic along the tree.

2.2 Guidelines and Limitations for VXLAN

VXLAN has the following guidelines and limitations:

- VxLAN and Fabric Path features cannot be enabled on the same VDC.
- 4 VTEP interfaces per VDC are supported. The total number of remote VTEPs is 1K per VDC.
- 1K VNIs are allowed per VDC.
- 1k BDIs are allowed per system.
- Bidir in underlay is not supported on vPC VXLAN Gateways.
- In VXLAN vPC deployments with F3 modules, the vPC peer-link should be on an isolated ASIC instance

from the Layer 3 core ports. After changing the peer-link ports to an isolated ASIC instance, ensure that you reload the switch.

- In a VXLAN vPC deployment with F3 modules and connected FEX devices, unknown unicast traffic from a remote source to the vPC switch pair may result in duplicate traffic if the MAC address is known to the vPC switches. Both vPC switches will receive a copy of the packet and forward it to the receiver behind FEX.

- In a VXLAN vPC deployment with F3 modules, known unicast traffic from a remote source to the vPC switch pair may result in loss of traffic if the MAC address is no longer known to the vPC switches.

- The number of IPv4 unicast routes supported is 64K shared with multicast routes.

- Number of IPv4 Multicast Groups is 32K due to software limitation.

- Maximum number of MAC addresses learned (local MACs and remote MACs) is 64K per F3 ASIC. (F3 MAC table size is 64K).

- VXLAN with IGMP snooping on VTEP tunnel interface is supported. You can configure ip igmp snooping disable-nve-static-router-port globally or per vlan to learn snooping states dynamically.

- VXLAN with Ingress replication using control plane is not supported.

- You can send Layer 3 end-to-end traffic with a maximum packet size of 9192 between VSI ports in different VNI segments.

- Layer 3 traffic will be dropped if the MTU packet size is greater than 9192.

- Inner VLAN class of service (CoS) needs to be propagated from ingress to egress in VXLAN. At the encapsulation side, CoS value will be copied to outer differentiated services code point (DSCP) and on the decapsulation side DSCP QoS will be copied back to egress VLAN COS.

- vPC peer-link can carry both global VLAN and bridge-domains that are having VXLAN enabled.

- MLD snooping on bridge-domain with VXLAN is not supported.

- ACL and QoS on a VTEP tunnel are not supported.

- Layer 3 VXLAN gateways are supported.

- PVLAN with VXLAN is not supported.

- Extending VLAN/VNI with OTV, VPLS, NV-GRE, and Layer 2 LISP when VXLAN is enabled is not supported due to hardware restrictions in the F3 ASIC.

- Extending Layer 2 MPLS network directly with VXLAN is not supported. Layer 2 MPLS has to be terminated and connected as a CE port for VXLAN extension due to forwarding restrictions.

- Interop with IPGRE is not supported.

- Interop with Layer 3 LISP is not supported.

- F3 SPAN feature does not support spanning L3 egress multicast packets.

- Netflow is not supported on VTEP interface.

- Netflow on VXLAN-enabled bridge domains is not supported.

- FEX ports are not supported as edge ports for VXLAN enabled bridge-domains.

- Extending global VLANs using VXLAN is not supported.

- BPDUs are not sent over the VTEP tunnel.

- Layer 3 Multicast - SSM in the core is not supported.

- MIB/XML support for VXLAN related changes is not supported.

- VXLAN encapsulation of an outer header with an IPv6 header is not supported.

- In VXLAN vPC Setup, RP should be configured on L3 core network. Direct connectivity between L2 gateway and vPC L3 gateway without L3 core in between is not supported.

- Any Source Multicast (ASM) is supported. Bidirectional PIM is supported on a single, non-vPC leaf switch.

- Physical port vPC for Vn-segment Service Instance (VSI) is not supported.

- The following Interface NVE counters are supported.

- Unicast and Multicast packets and bytes transmitted

- Unicast and Multicast packets and bytes received

- SPAN is not supported for NVE tunnel interfaces.
- Equal cost multipath (ECMP) on the core is based on inner packet (DMAC, SMAC) combination.
- Maximum 64 MST instances are supported.
- MST scale limit is 300K logical ports on unidimensional setup.
- When vPC peer-link flap, Mac addresses will be flushed for orphan and vPC ports. End point needs to re-ARP.
- 32 VSI encapsulation profiles are supported per interface.
- SSM is not supported
- Bi-Dir multicast mode is not supported for vPC
- IGMP Snooping Layer2 Multicast Mac lookup mode is not supported.
- IPv6 for Multicast is not supported.
- Hypervisor VLANs can be configured using regular VLAN and trunk or access port configurations.
- vPC peer-link can carry both global VLANs and bridge-domains that are VXLAN enabled.

2.2.1 Considerations for VXLAN Deployment

- A loopback address is required when using the source-interface config command. The loopback address represents the local VTEP IP.
 - For the Network Virtualization Endpoint (NVE) source loop back address, secondary address should be same on both the vPC peers and the primary address should be different.
 - To establish IP multicast routing in the core, IP multicast configuration, PIM configuration, and RP configuration is required.
 - VTEP to VTEP unicast reachability can be configured through any IGP protocol.

2.2.2 vPC Considerations for VXLAN Deployment

- Bind NVE to a loopback address that is separate from other loopback addresses that are required by Layer 3 protocols. A best practice is to use a dedicated loopback address for VXLAN.
- The loopback address used by NVE needs to be configured to have a primary IP address and a secondary IP address.
 - The secondary IP address is used for all VXLAN traffic that includes multicast and unicast encapsulated traffic.
- vPC peers must have identical configurations as listed below:
 - Consistent Bridge-domain to VNI mapping.
 - Consistent NVE binding to the same loopback interface.
 - The same secondary IP address must be configured on both the switches. However, each vPC peer switch should have a unique primary IP address assigned to it.
 - Consistent NVE interface to VNI mapping.
 - Consistent VNI to group mapping.
- For multicast, the vPC node that receives the (S, G) join from the RP (rendezvous point) becomes the DF (designated forwarder). On the DF node, encap routes are installed for multicast.

Decap routes are installed based on the election of a decapper from between the VPC primary node and the VPC secondary node. The winner of the decap election is the node with the least cost to the RP or the source. However, if the cost to the RP is the same for both nodes, the VPC primary node is elected.

The winner of the decap election has the decap mroute installed. The other node does not have a decap route installed.

- On a VPC device, BUM traffic (broadcast, unknown-unicast, and multicast traffic) from hosts is replicated on the vPC peer-link. A copy is made of every native packet and each native packet is sent across the vPC peer-link to service orphan-ports connected to the peer VPC switch.

To prevent traffic loops in VXLAN networks, native packets ingressing the vPC peer-link cannot be sent to an uplink. However, if the peer switch is the encapper, the copied packet traverses the vPC peer-link and is sent to the uplink.

In a VXLAN vPC deployment with peer switch, encapsulation profile, and bridge domain configurations, the vPC secondary peer switch does not generate or process BPDUs for bridge domains.

- When vPC peer-link is shut, the loopback primary address is used as the source IP address for encapsulation by both vPC switches.

- When vPC peer-link is no-shut, the NVE loopback secondary address is used.

- For VPC, the loopback interface has 2 IP addresses: the primary IP address and the secondary IP address. The primary IP address is unique and is used by Layer 3 protocols.

The secondary IP address on loopback is necessary because the interface NVE uses it for the VTEP IP address. The secondary IP address must be same on both vPC peers.

- The VPC peer-gateway feature must be enabled on both peers. As a best practice, use peer-switch, peer gateway, ip arp sync, ipv6 nd sync configurations for improved convergence in VPC topologies.

The following is an example (best practice) of a VPC configuration:

```
switch# sh ru vpc

version 6.1(2)I3(1) feature vpc
vpc domain 2 peer-switch
  peer-keepalive destination 172.29.206.65 source
  172.29.206.64 peer-gateway
  ipv6 nd
  synchron
  ize ip
  arp
  synchron
  ize
```

- On a VPC pair, shutting down NVE or NVE loopback on one of the VPC nodes is not a supported configuration. This means that traffic fail over on one-side NVE shut or one-side loopback shut is not supported.

- Redundant anycast RPs configured in the network for multicast load-balancing and RP redundancy are supported on VPC VTEP topologies.

- The following are the examples of SVI with PIM enabled:

```
switch# show run interface BDI3

interface BDI3
  description
  special_svi_over_mct
  no shutdown

  ip address
  30.2.1.1/30 ip
  pim sparse-
  mode

switch# show run interface BDI3

interface BDI3
  description special_svi_over_vPC peer-link no
  shutdown ipv6 address FE80::290:27FF:FE8C:B709
  ip pim sparse-mode
```

- As a best practice when changing the secondary IP address of an anycast VPC VTEP, the NVE interfaces on

both the VPC primary and the VPC secondary should be shut before the IP changes are made.

- For a VXLAN vPC deployment, you should configure the switchport trunk native vlan tag exclude control command on the interface port channel configured as the vPC peer-link.

2.2.3 Network Considerations for VXLAN Deployments

- MTU Size in the Transport Network

Due to the MAC-to-UDP encapsulation, VXLAN introduces 50-byte overhead to the original frames. Therefore, the maximum transmission unit (MTU) in the transport network needs to be increased by 50 bytes. If the overlays use a 1500-byte MTU, the transport network needs to be configured to accommodate 1550-byte packets at a minimum. Jumbo-frame support in the transport network is required if the overlay applications tend to use larger frame sizes than 1500 bytes.

- ECMP and LACP Hashing Algorithms in the Transport Network

As described in a previous section, Inspur CN12700 Series Switches introduce a level of entropy in the source UDP port for ECMP and LACP hashing in the transport network. As a way to augment this implementation, the transport network uses an ECMP or LACP hashing algorithm that takes the UDP source port as an input for hashing, which achieves the best load-sharing results for VXLAN encapsulated traffic.

- Multicast Group Scaling

The VXLAN implementation on Inspur CN12700 Series Switches uses multicast tunnels for broadcast, unknown unicast, and multicast traffic forwarding. Ideally, one VXLAN segment mapping to one IP multicast group is the way to provide the optimal multicast forwarding. It is possible, however, to have multiple VXLAN segments share a single IP multicast group in the core network. VXLAN can support up to 16 million logical Layer 2 segments, using the 24-bit VNID field in the header. With one-to-one mapping between VXLAN segments and IP multicast groups, an increase in the number of VXLAN segments causes a parallel increase in the required multicast address space and the amount of forwarding states on the core network devices. At some point, multicast scalability in the transport network can become a concern. In this case, mapping multiple VXLAN segments to a single multicast group can help conserve multicast control plane resources on the core devices and achieve the desired VXLAN scalability. However, this mapping comes at the cost of suboptimal multicast forwarding. Packets forwarded to the multicast group for one tenant are now sent to the VTEPs of other tenants that are sharing the same multicast group. This causes inefficient utilization of multicast data plane resources. Therefore, this solution is a trade-off between control plane scalability and data plane efficiency.

Despite the suboptimal multicast replication and forwarding, having multiple-tenant VXLAN networks to share a multicast group does not bring any implications to the Layer 2 isolation between the tenant networks. After receiving an encapsulated packet from the multicast group, a VTEP checks and validates the VNID in the VXLAN header of the packet. The VTEP discards the packet if the VNID is unknown to it. Only when the VNID matches one of the VTEP's local VXLAN VNIDs, does it forward the packet to that VXLAN segment. Other tenant networks will not receive the packet. Thus, the segregation between VXLAN segments is not compromised.

- Spanning Tree Protocol (STP) domain

Ensure that the root of an STP domain local to the VXLAN fabric is a VTEP, or placed within the fabric. The STP root should not be outside the VXLAN fabric (below the VTEPs) since it will lead to Layer 2 loops.

2.2.4 Considerations for the core VRF

The following are considerations for the configuration of the core VRF:

- On the VTEP device:

- Enable and configure IP multicast.
- Create and configure a loopback interface with a /32 IP address.
- Enable IP multicast on the loopback interface.
- Advertise the loopback interface/32 addresses through the routing protocol that runs in the transport network.

- Enable IP multicast on the uplink outgoing physical interface.
- Throughout the transport network:
 - Enable and configure IP multicast.

2.2.5 ISSU Support

The following are the ISSU support details for VXLAN flood and learn deployment:

- Inspur CN12700 Series switches running Inspur INOS Release 8.4(1).
- Virtual Device Context (VDC) Types:
 - F3 Only

Supported ISSU Steps:

1. Upgrade ISSU to Inspur INOS Release 8.4(1).
2. For F3 Only VDC, configure default interface for all L2/L3 interfaces.
3. Reload F3 Only VDC or the switch.
4. Enable “feature nve” in F3 Only VDC.
5. Configure VXLAN - VSI/NVE.

2.3 Configuring VXLAN

2.3.1 Enabling VXLANs

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	[no] feature nv overlay	Enables the VXLAN feature.
Step 3	[no] feature vni	Configures the global mode for all VXLAN bridge domains.
Step 4	[no] vni [range]	Defines the VNI range.
Step 5	(Optional) copy running-config startup-config	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

Example

```
switch# configure terminal
switch(config)# feature nv
overlay      switch(config)#
feature          vni
switch(config)# vni 7000
switch(config)# copy running-config startup-config
```

2.3.2 Configuring VNI Service Instances

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	encapsulation profile vni <i>vsi-range</i>	Encapsulates traffic in the virtual network.
Step 3	dot1q var vni <i>vnid</i>	Defines the matching criteria to map Q-in-Q ingress frames.
Step 4	interface <i>var</i>	Enters interface configuration mode.
Step 5	service instance <i>num vni</i>	Creates a service instance.
Step 6	no shut down	Enables the service instance on this interface.
Step 7	encapsulation profile <i>vsi-range</i>	Encapsulates traffic in the virtual network.

Example

The following example shows how to configure VNI service instances:

```

config t
    encapsulation profile vni
        vsi_100_to_7000 dot1q 100
        vni 7000
    interface e1/1
        service
        instance 1
        vni no shut
        down
        encapsulation profile vsi_100_to_7000

```

2.3.3 Mapping VLAN to VNI

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	system bridge-domain <i>ID</i>	Identify the bridge domain IDs that are available for bridge-domain configuration.
Step 3	bridge-domain <i>vlan-ID</i>	Enables bridging to map VLAN to VXLAN VNI.
Step 4	member vni <i>number</i>	Maps VXLAN to a bridge domain.

Example

The following example shows how to map an encapsulation profile to a VNI:

```

switch# configure terminal
switch(config)# system bridge-domain 100-500
switch(config)# bridge-domain 100
switch(config)# member vni 7000

```

2.3.4 Creating an VTEP and NVE Interface

An NVE interface is the overlay interface that terminates VXLAN tunnels.

There is a one-on-one mapping between NVE interface configuration and the source interface. Source interface used under a NVE cannot be reused.

You can create and configure an NVE (overlay) interface with the following:

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	interface nve <i>x</i>	Creates a VXLAN overlay interface that terminates VXLAN tunnels. Note Only 4 NVE interfaces are allowed on the switch. Range is from 1 to 4.
Step 3	source-interface loopback <i>src-if</i>	The source interface must be a loopback interface that is configured on the switch with a valid /32 IP address. This /32 IP address must be known by the transient devices in the transport network and the remote VTEPs. This is accomplished by advertising it through a dynamic routing protocol in the transport network.
Step 4	member vni <i>vni</i> [mcast-group <i>mcast-ip</i>]	Associate VXLAN VNIs (Virtual Network Identifiers) with the NVE interface.
Step 5	exit	Exits interface configuration mode.
Step 6	interface loopback <i>if-number</i>	Creates a loopback interface.
Step 7	ip address <i>address</i>	Assigns an ip address to the configured interface.
Step 8	vrf member core	Creates a vrf member core in the interface.

Example

The following example shows how to create a VTEP / NVE interface:

```
switch# configure terminal
switch(config)# interface nve 1
switch(config-if)# source-interface loopback 10
switch(config-if)# member vni 7000 mcast-group 225.1.1.1
switch(config-if)# member vni 8000 mcast-group 226.1.1.1
switch# configure terminal
switch(config)# interface loopback 10
switch(config-if)# ip address 10.1.1.1/32
switch(config-if)# vrf member core
```

2.3.5 Configuring vPC Peer-link

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	interface var	Enters interface configuration mode.
Step 3	switchport	Sets the interface type to be a Layer 2 host port.
Step 4	switchport mode trunk	Sets the interface as a Layer 2 trunk port.
Step 5	switchport trunk native vlan tag exclude control	Enables native VLAN tagging on a trunk port, while ensuring that the control packets on the native VLAN are untagged. Note For trunk ports, by default data and control packets are untagged. The switchport trunk native vlan tag command form ensures that control and data packets of the native VLAN are tagged.
Step 6	spanning-tree port type network	Configures the interface that connects to a Layer 2 switch or bridge as a network spanning tree port.
Step 7	vpc peer-link	Configures the port channel as a vPC peer-link.

Example

The following example shows how to configure vPC peer-link:

```
switch# configure terminal
switch(config)# interface port-channel10
switch(config-if)# switchport
switch(config-if)# switchport mode trunk
switch(config-if)# switchport trunk native vlan tag exclude control
switch(config-if)# spanning-tree port type network
switch(config-if)# vpc peer-link
```

2.3.6 Configuring L3 Interface on VXLAN Tunnel/Hypervisor VLAN

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	interface var	Enters interface configuration mode.

Step 3	ip address <i>address</i>	Configures the IP address on the interface.
Step 4	vrf member <i>core</i>	Creates a vrf member core in the interface.

Example

The following example shows how to configure L3 interface on VXLAN tunnel/hypervisor VLAN:

```
switch#      configure      terminal
switch(config)#      interface      vlan      300
switch(config-if)#   ip      address  11.1.1.1/24
switch(config-if)#   vrf      member  core
```

2.3.7 Configuring L3 Interface for IP Cloud Connectivity

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	interface <i>var</i>	Enters interface configuration mode.
Step 3	no switchport	Disables the switchport.
Step 4	ip address <i>address</i>	Configures the IP address on the interface.
Step 5	vrf member <i>core</i>	Creates a vrf member core in the interface.

Example

The following example shows how to configure L3 interface for IP cloud connectivity:

```
switch#      configure      terminal
switch(config)#      interface      e7/1
switch(config-if)#   no      switchport
switch(config-if)#   ip      address  11.1.1.1/24
switch(config-if)#   vrf      member  core
```

2.3.8 Configuring L3 Interface on Tenant Bridge-Domains/VNIs in L3 Gateway

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	feature interface-vlan	Enables the creation of BDI interfaces.
Step 3	interface bridge-domain <i>number</i>	Enters interface configuration mode.
Step 4	ip address <i>address</i>	Configures the IP address on the interface.

Step 5	vrf member <i>tenant</i>	Configures the VRF member.
Step 6	hsrp <i>var</i>	Creates an HSRP group and enters HSRP configuration mode.
Step 7	ip address <i>address</i>	Configures the virtual IP address for the HSRP group and enables the group. Repeat steps 1-5 to configure another bridge-domain interface and HSRP.

Example

The following example shows how to configure L3 interface on tenant bridge-domains/VNIs in L3 gateway:

```

feature
interface-vlan
interface
bridge-domain
100
    ip address
    50.1.1.2/24
    vrf member
    tenant
    hsrp 50
        ip address 50.1.1.1

feature
interface-vlan
interface
bridge-domain
200
    ip address
    60.1.1.2/24
    vrf member
    tenant
    hsrp 60
        ip address 60.1.1.1

```

2.3.9 Disabling VXLANs

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters configuration mode.
Step 2	no feature vni	Disables the global mode for all VXLAN bridge domains
Step 3	no feature nv overlay	Disables the VXLAN feature.
Step 4	no bridge-domain	Disables bridging to map VLAN.
Step 5	no member vni	Dissociates VXLAN VNIs from the NVE interface.

Step 6	no vni	Removes the VXLAN segment ID to which the VLAN is mapped.
Step 7	(Optional) copy running-config startup-config	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

2.4 Verifying the VXLAN Configuration

To display the VXLAN configuration information, enter one of the following commands:

Command	Purpose
show tech-support vxlan	Displays related VXLAN tech-support information.
show logging level nve	Displays logging level.
show tech-support nve	Displays related NVE tech-support information.
show run interface nve x	Displays NVE overlay interface configuration.
show nve interface nve x	Displays NVE overlay interface status.
show interface nve x	Displays all the counters of an NVE interface.
show nve peers	Displays NVE peer status.
show bridge-domain	Displays the bridge domain information.
show run vni	Displays the VXLAN VNI configuration.
show interface nve counters	Displays the NVE counters in an interface.
show interface bdi	Displays the configuration summary of the corresponding BDI.
show vni x	Displays the list of all VNIs
show ip mroute	Displays information about mroute entries in the mroute table.
show nve VXLAN-params	Displays VXLAN parameters, such as VXLAN destination or UDP port.
show nve internal platform interface nve 1 detail	Displays NVE overlay internal detailed information.
show nve vxlan-params	Displays VXLAN parameters, such as VXLAN destination or UDP port.

2.5 Configuration Examples

2.5.1 Example of VXLAN Flood and Learn Configuration

The following example shows the VXLAN Flood and Learn configuration.

VTEP-1:

```

feature pim
system bridge-
domain 50,75
feature nv
overlay feature
interface-vlan

feature vni vni 30000
vni 50000

ip route 10.10.10.2/32 Ethernet10/1 10.1.1.2
ip pim rp-address 10.1.1.1 group-list
209.165.0.0/4

bridge-domain 50
bridge-domain 75

encapsulation profile vni
VSI_50_TO_5000 dot1q 50
vni 5000
encapsulation profile vni
VSI_75_TO_7500 dot1q 75
vni 7500
bridge-domain 50
member vni 5000
bridge-domain 75
member vni 7500

interface nve1
no shutdown
source-interface loopback10
member vni 5000 mcast-group 209.165.1.1
member vni 7500 mcast-group 209.165.1.1

interface Bdi50
no shutdown
ip address 10.50.50.50/24

interface Bdi75
no shutdown
ip address 10.75.75.75/24

interface
Ethernet7
/17 no
switchpor
t
no shutdown
service
instance
1 vni no
shutdown
encapsulation default profile
VSI_50_TO_5000 service
instance 2 vni
no shutdown
encapsulation profile VSI_75_TO_7500
default

interface
Ethernet1
0/1 no

```

```

switchport
t
ip address 10.1.1.1/30

ip pim
sparse-
mode no
shutdown

interface loopback10
ip address
10.10.10.1/32 ip
pim sparse-mode

```

VTEP-2:

```

feature pim
system bridge-
domain 50,75
feature nv
overlay feature
interface-vlan

feature vni vni 32000
vni 52000

ip route 10.10.10.1/32 Ethernet10/7 10.1.1.1
ip pim rp-address 10.1.1.1 group-list 209.165.0.0/4

bridge-domain 50
bridge-domain 75

encapsulation profile vni
VSI_50_TO_5000 dot1q 50 vni
5000
encapsulation profile vni
VSI_75_TO_7500 dot1q 75 vni
7500

bridge-domain 50
member vni 5000
bridge-domain 75
member vni 7500

interface nve1 no shutdown
source-interface loopback10
member vni 5000 mcast-group 209.165.1.1
member vni 7500 mcast-group 209.165.1.1

interface Bdi50 no shutdown
ip address 10.50.50.51/24

interface Bdi75 no shutdown
ip address 10.75.75.76/24

interface
Ethernet7/30
no
switchport
no shutdown
service
instance 1
vni no
shutdown
encapsulation profile VSI_50_TO_5000 default

```

```

service
  instance
    2 vni no
  shutdown
  encapsulation profile VSI_75_TO_7500
  default

interface
  Ethernet1
    0/7 no
  switchport
  ip address
    10.1.1.2/30
  ip pim
  sparse-mode
  no shutdown

interface loopback10
  ip address
    10.10.10.2/32
  ip pim
  sparse-mode

```

2.5.2 Example of Verifying VXLAN Flood and Learn Configuration

The following example shows the VXLAN Flood and Learn configuration verification.

VTEP-1:

```

VTEP-1# show nve vni
Codes: CP - Control Plane      DP -
       Data Plane UC - Unconfigured
       SA -
       Suppress ARP

Interface VNI      Multicast-group  State Mode Type [BD/VRF]      Flags
-----
nve1  5000          209.165.1.1     Up   DP  L2 [50]
nve1  7500          192.168.1.1     Up   DP  L2 [75]

VTEP-1# show running-config interface nve 1

interface nve1 no shutdown
 source-interface loopback10
 member vni 5000 mcast-group 209.165.1.1
 member vni 7500 mcast-group 192.168.1.1

VTEP-1# show service instance

vni detail VSI: VSI-

Ethernet7/17.1
If-index:
0x353100
01 Admin
Status:
Up Oper
Status:
Up
Auto-configuration Mode: No
encapsulation profile vni
VSI_50_TO_5000

```

```

    dot1q
50 vni
5000
Dot1q VNI BD
-----
-----
-----
50 5000 50

VSI: VSI-
Ethernet7/1
7.2 If-
index:
0x35310002
Admin
Status: Up
Oper Status: Up
Auto-configuration
Mode: No
encapsulation
profile vni TEST
    dot1q
100 vni
7500
Dot1q

VNI

BD
-----
-----
-----
100 7500 75

VTEP-1# show bridge-domain

Bridge-domain 50 (2 ports
in all) Name:: Bridge-
Domain50

Administrative State: UP
Operational State: UP
VSI-Eth7/17.1
vni5000 nvel

Bridge-domain 75 (2 ports
in all) Name:: Bridge-
Domain75

Administrative State: UP
Operational State: UP
VSI-Eth7/17.2
vni7500 nvel

VTEP-1# 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
-----+-----+-----+-----+-----+-----+-----+-----
* 50   547f.eeec.af43   dynamic   ~~~      F      F      nve1/10.10.10.2
* 50   547f.eeec.af44   dynamic   ~~~      F      F      VSI-Eth7/17.1
* 50   547f.eeec.af45   dynamic   ~~~      F      F      nve1/10.10.10.2
* 75   547f.eeec.af44   dynamic   ~~~      F      F      VSI-Eth7/17.2
* 75   547f.eeec.af45   dynamic   ~~~      F      F      nve1/10.10.10.2

VTEP-1# show ip mroute detail
IP Multicast Routing Table for VRF "default"

Total number of routes:
7 Total number of (*,G)
routes: 2 Total number
of (S,G) routes: 4
Total number of (*,G-prefix) routes: 1

(*, 209.165.1.1/32), uptime: 19:51:28, nve(1) ip(0)
  pim(1) Data Created: No
VXLAN Flags VXLAN Encap
  Stats: 0/0 [Packets/Bytes], 0.000 bps
  Incoming interface: Ethernet10/1, RPF nbr:
  1.1.1.1 Outgoing interface list: (count: 2)
    Ethernet10/1, uptime: 19:51:09,
    pim, (RPF) nve1, uptime: 19:51:28,
    nve

(10.10.10.1/32, 209.165.1.1/32), uptime: 19:51:28, nve(0) mrib(0) ip(0) pim(1)
  Data Created:
  No Received
  Register stop
  VXLAN Flags
    VXLAN Encap
    Stats: 19/2274 [Packets/Bytes], 0.000 bps
    Incoming interface: loopback10, RPF nbr: 10.10.10.1,
    internal Outgoing interface list: (count: 1)
      Ethernet10/1, uptime: 19:51:09, pim

(10.10.10.2/32, 209.165.1.1/32), uptime: 18:10:06, pim(1) mrib(1) ip(0)
  Data
  Created:
  Yes VXLAN
  Flags

    VXLAN Decap
    Stats: 9/846 [Packets/Bytes], 0.000 bps
    Incoming interface: Ethernet10/1, RPF nbr:
    1.1.1.2, internal Outgoing interface list: (count:
    2)
      Ethernet10/1, uptime: 01:00:32,
      pim, (RPF) nve1, uptime:
      18:10:06, mrib

(*, 209.165.1.1/32), uptime: 12:52:13, nve(1)
  ip(0) pim(1) Data Created: No
  VXLAN Flags VXLAN Encap
  Stats: 0/0 [Packets/Bytes], 0.000 bps
  Incoming interface: Ethernet10/1, RPF
  nbr: 1.1.1.1 Outgoing interface list:
  (count: 2)
    Ethernet10/1, uptime: 12:51:52,

```



```

    pim, (RPF) nve1, uptime:
    12:52:13, nve

(10.10.10.1/32, 209.165.1.1/32), uptime: 12:52:13, nve(0) mrib(0) ip(0) pim(1)
Data
Created: No
Received
Register
stop VXLAN
Flags
  VXLAN Encap
Stats: 300/39850 [Packets/Bytes], 0.000
      b
ps
Incoming interface: loopback10, RPF nbr:
10.10.10.1, internal Outgoing interface list:
(count: 1)
  Ethernet10/1, uptime: 12:51:52, pim

(10.10.10.2/32, 209.165.1.1/32), uptime: 12:51:34, pim(1) mrib(1) ip(0)
Data
Create
d: Yes
VXLAN
Flags
  VXLAN Decap
Stats: 22/1928 [Packets/Bytes], 0.000 bps
Incoming interface: Ethernet10/1, RPF nbr:
1.1.1.2, internal Outgoing interface list: (count:
2)
  Ethernet10/1, uptime: 00:52:14,
  pim, (RPF) nve1, uptime:
  12:51:34, mrib

(*, 209.166.0.0/8), uptime: 20:56:33, pim(0)
ip(0)
Data Created: No
Stats: 0/0 [Packets/Bytes],
0.000bps Incoming interface:
Null, RPF nbr: 0.0.0.0
Outgoing interface list:
(count: 0)

VTEP-1# show ip arp

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 default Total
number of entries: 4
Address Age MAC Address Interface
10.50.50.1 00:11:32 547f.eeec.af44 Bdi50
10.50.50.2 00:11:14 547f.eeec.af44 Bdi50
10.75.75.1 00:10:45 547f.eeec.af44 Bdi75
10.75.75.2 00:15:04 547f.eeec.af45 Bdi75
10.1.1.2 00:05:39 547f.eeec.af43 Ethernet10/1

VTEP-1# show ip route
IP Route Table for VRF
"default" '*' denotes

```

```

best ucast next-hop
'***' denotes best mcast
next-hop '[x/y]'
denotes
[preference/metric]

'%<string>' in via output denotes VRF <string>

10.1.1.0/30, ubest/mbest: 1/0, attached
  *via 10.1.1.1, Eth10/1, [0/0], 20:25:13,
direct 1.1.1.1/32, ubest/mbest: 1/0,
attached
  *via 10.1.1.1, Eth10/1, [0/0],
20:25:13, local 10.10.10.1/32, ubest/mbest:
2/0, attached
  *via 10.10.10.1, Lo10, [0/0], 20:25:45, local
  *via 10.10.10.1, Lo10, [0/0], 20:25:45, direct
10.10.10.2/32, ubest/mbest: 1/0
  *via 10.1.1.2, Eth10/1, [1/0], 20:23:42,
static 10.50.50.0/24, ubest/mbest: 1/0,
attached
  *via 10.50.50.50, Bdi50, [0/0], 01:18:47,
direct 10.50.50.50/32, ubest/mbest: 1/0,
attached
  *via 10.50.50.50, Bdi50, [0/0], 01:18:47, local
10.75.75.0/24, ubest/mbest: 1/0, attached
  *via 10.75.75.75, Bdi75, [0/0], 01:10:05,
direct 10.75.75.75/32, ubest/mbest: 1/0,
attached
  *via 10.75.75.75, Bdi75, [0/0], 01:10:05, local

```

VTEP-2:

```

VTEP-2# show nve vni
Codes: CP - Control Plane      DP - Data
       Plane UC - Unconfigured SA -
       Suppress ARP

Interface VNI      Multicast-group  State Mode Type [BD/VRF]      Flags
-----
nve1  5000          209.166.1.1     Up   DP  L2 [50]
nve1  7500          192.168.1.1     Up   DP  L2 [75]

VTEP-2# show running-config interface nve 1

interface nve1 no shutdown
source-interface loopback10
member vni 5000 mcast-group 209.166.1.1
member vni 7500 mcast-group

192.168.1.1 VTEP-2# show service

instance vni detail

VSI:      VSI-
Ethernet7/30.1
If-index:
0x3531d001
Admin Status:
Up
Oper Status: Up
Auto-configuration Mode: No
encapsulation profile vni
VSI_50_TO_5000
dot1q 50

```

```
vni 5000
Dot1q VNI BD
-----
-----
50
      50
00    50
```

```
VSI: VSI-
Ethernet7/30.2
If-index:
0x3531d002
Admin Status:
Up
Oper Status: Up
Auto-configuration
Mode: No encapsulation
profile vni TEST
  dot1q 100
  vni 7500
  Dot1q VNI BD
  -----
  -----
100
      75
00    75
```

VTEP-2# show bridge-domain

```
Bridge-domain 50 (2 ports
in all) Name:: Bridge-
Domain50
Administrative State: UP
Operation
      al State: UP vni5000
      VSI-Eth7/30.1
      nvel
```

```
Bridge-domain 75 (2 ports
in all) Name:: Bridge-
Domain75
Administrative State: UP
Operation
      al State: UP vni7500
      VSI-Eth7/30.2
      nvel
```

VTEP-2# 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
-----+-----+-----+-----+-----+-----
* 50 547f.eeec.af44 dynamic ~~~ F F nvel/10.10.10.1
```

```
* 50 547f.eeec.af45 dynamic ~~~ F F VSI-Eth7/30.1
* 75 547f.eeec.af45 dynamic ~~~ F F VSI-Eth7/30.2
* 75 547f.eeec.af48 dynamic ~~~ F F nve1/10.10.10.1
```

VTEP-2# show ip mroute detail

IP Multicast Routing Table for VRF "default"

Total number of

routes: 5 Total

number of (*,G)

routes: 2 Total

number of (S,G)

routes: 2

Total number of (*,G-prefix) routes: 1

(*, 209.165.1.1/32), uptime: 19:56:19, nve(1)

ip(0) pim(0) Data Created: No

VXLAN Flags VXLAN

Encap

Stats: 8/748 [Packets/Bytes], 0.000

bps Incoming interface: Ethernet10/7,

RPF nbr: 1.1.1.1 Outgoing interface

list: (count: 1)

nve1, uptime: 19:56:19, nve

(10.10.10.2/32, 209.165.1.1/32), uptime: 19:56:19, nve(0) mrib(0) pim(1) ip(0)

Data

Created: No

Received

Register

stop VXLAN

Flags

VXLAN Encap

Stats: 9/834 [Packets/Bytes], 0.000 bps

Incoming interface: loopback10, RPF nbr:

10.10.10.2 Outgoing interface list:

(count: 1)

Ethernet10/7, uptime: 18:15:17, pim

(*, 209.165.1.1/32), uptime: 12:57:03, nve(1)

ip(0) pim(0) Data Created: No

VXLAN Flags VXLAN Encap

Stats: 10/864 [Packets/Bytes], 0.000 bps

Incoming interface: Ethernet10/7, RPF nbr:

1.1.1.1 Outgoing interface list: (count: 1)

nve1, uptime: 12:57:03, nve

(10.10.10.2/32, 209.165.1.1/32), uptime: 12:57:03, nve(0) mrib(0) ip(0) pim(1)

Data Created:

No Received

Register stop

VXLAN Flags

VXLAN Encap

Stats: 30/2648 [Packets/Bytes], 0.000 bps

Incoming interface: loopback10, RPF nbr:

10.10.10.2 Outgoing interface list: (count:

1)

Ethernet10/7, uptime: 12:56:45, pim

(*, 209.167.0.0/8), uptime: 18:20:36, pim(0) ip(0)

Data Created: No

Stats: 0/0 [Packets/Bytes], 0.000 bps Incoming interface: Null, RPF nbr: 0.0.0.0 Outgoing i

```

VTEP-2# show ip arp

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
default Total number of
entries: 4
Address      Age      MAC Address  Interface
10.50.50.1   00:11:30  547f.eeec.af44 Bdi50
10.50.50.2   00:17:07  547f.eeec.af45 Bdi50
10.75.75.1   00:04:14  547f.eeec.af45 Bdi75
10.75.75.2   00:03:24  547f.eeec.af45 Bdi75
10.1.1.1     00:10:52  547f.eeec.af48 Ethernet10/7

VTEP-2# show ip route
IP Route Table for VRF
"default" '*' denotes best
ucast next-hop '*' denotes
best mcast next-hop '[x/y]'
denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.1.1.0/30, ubest/mbest: 1/0, attached
  *via 10.1.1.2, Eth10/7, [0/0], 20:30:24,
  direct 10.1.1.2/32, ubest/mbest: 1/0,
  attached
  *via 10.1.1.2, Eth10/7, [0/0],
  20:30:24, local 10.10.10.1/32, ubest/mbest:
  1/0
  *via 10.1.1.1, Eth10/7, [1/0], 20:29:48,
  static 10.10.10.2/32, ubest/mbest: 2/0,
  attached
  *via 10.10.10.2, Lo10, [0/0], 20:29:39, local
  *via 10.10.10.2, Lo10, [0/0], 20:29:39, direct
10.50.50.0/24, ubest/mbest: 1/0, attached
  *via 10.50.50.51, Bdi50, [0/0], 01:22:50,
  direct 10.50.50.51/32, ubest/mbest: 1/0,
  attached
  *via 10.50.50.51, Bdi50, [0/0], 01:22:50, local
10.75.75.1/24, ubest/mbest: 1/0, attached
  *via 10.75.75.76, Bdi75, [0/0], 01:14:50,
  direct 10.75.75.76/32, ubest/mbest: 1/0,
  attached
  *via 10.75.75.76, Bdi75, [0/0], 01:14:50, local

```

2.6 Feature History for VXLAN Flood and Learn

This table lists the release history for this feature.

Table 1 : Feature History for VXLAN Flood and Learn

Feature Name	Releases	Feature Information	
VXLAN Flood and Learn	8.4(1)	This feature was introduced.	

CHAPTER 3 Configuring VXLAN BGP EVPN

This chapter contains the following sections:

- Information About VXLAN BGP EVPN.
- Configuring VXLAN BGP EVPN.
- Feature History for VXLAN BGP EVPN.

3.1 Information About VXLAN BGP EVPN

3.1.1 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.

VXLAN as the overlay technology

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 and 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:

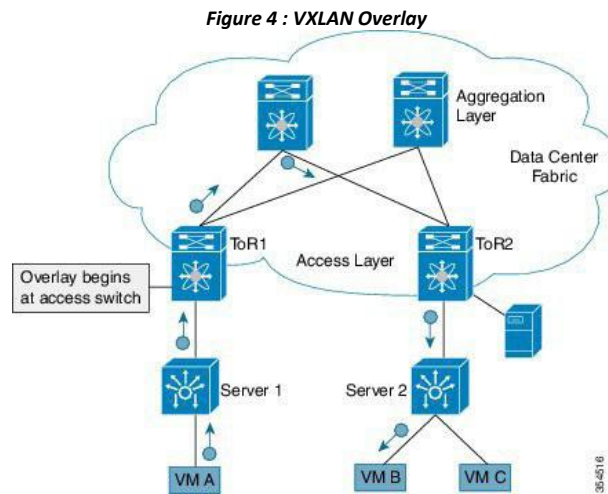
The example below shows why spine switch table sizes are not increased due to VXLAN fabric overlay, making them lean.

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.



- 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 ToR2 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 and 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.

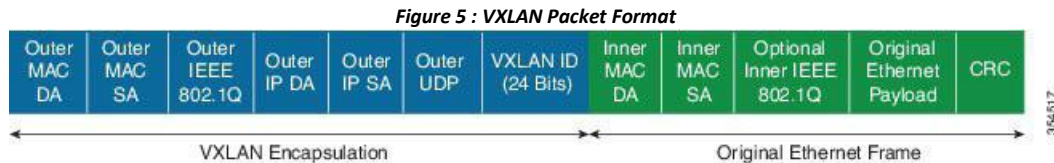
- In order to accurately route/switch packets between end hosts in the data center, each participating ToR in a VXLAN cluster must be aware of the end hosts attached to it and also the end hosts attached to other ToRs, in real time.

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.

3.1.2 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.



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.

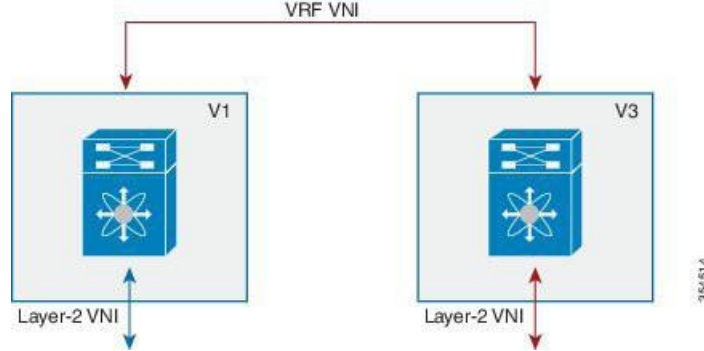
- By confining server and end host identification of a specific tenant to a unique VNI (or few unique VNIs), segmentation and security are ensured.

- 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 6 : 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.

3.1.3 Fabric Overlay Control-Plane (MP-BGP EVPN)

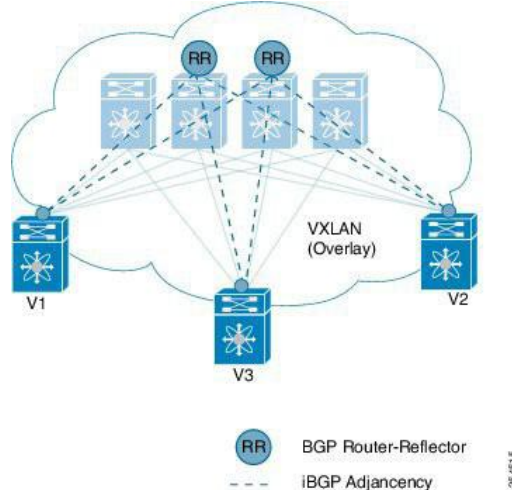
The main reasons for using BGP EVPN as the overlay control plane are:

- *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 7 : End Host IP + MAC Address Distribution in a VXLAN Setup



The reasons 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 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*

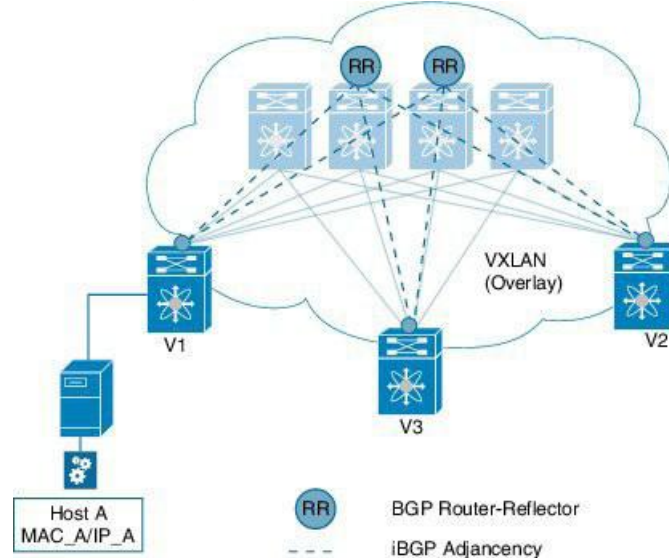
In a VXLAN-EVPN fabric, traffic is only accepted from VTEPs whose information is learnt via the BGP-EVPN control plane. Any VXLAN encapsulated traffic received from a VTEP that is not known via the control plane will be dropped. In this way, this presents a secure fabric where traffic will only be forwarded 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.

3.1.4 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 8 : 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.

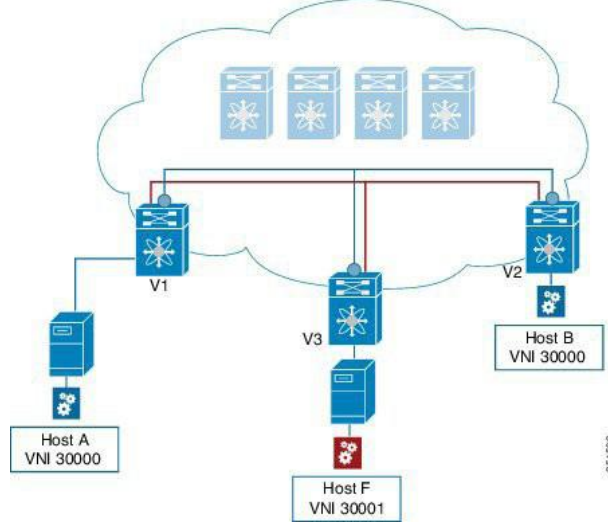
The gateway provides routing and bridging capabilities, and the mechanism is referred as *Integrated Routing and Bridging* (IRB).

The distributed anycast gateway for routing is completely stateless and does not require the exchange of protocol signaling 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 is given below:

Figure 9 : Distributed Gateway



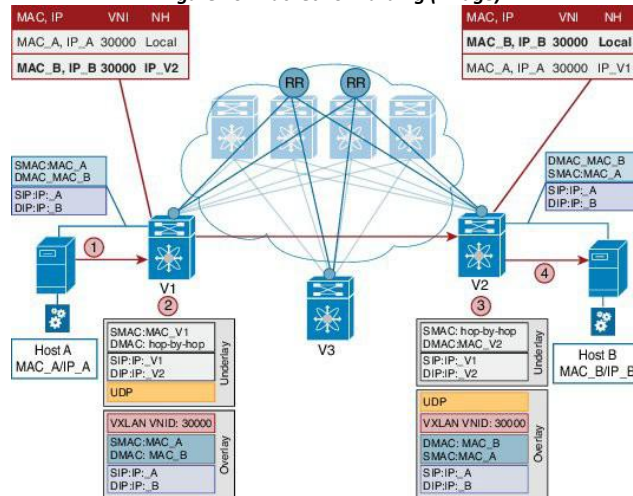
Blue tenant network's configuration

Similar configuration needs to be implemented on the leaf switch V1 and all other switches containing the blue tenant network's end hosts. Configuration details are provided in BGP EVPN and Overlay Configuration chapter. Similar configuration needs to be implemented for red tenant too.

After the configurations, 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 10 : Packet Forwarding (Bridge)

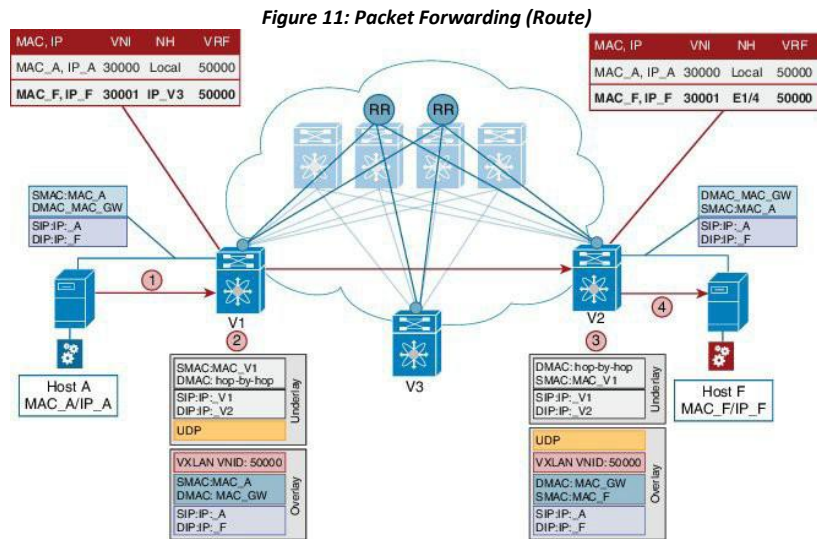


The VNI of the source end host, Host A, and the target end host, Host B, is 30000.

- Host A sends traffic to the directly attached VTEP V1.
- 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).
- VTEP V1 bridges the packets and sends it toward VTEP V2 with a VXLAN header stamped with the Layer 2 VNI 30000.
- 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



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.

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

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.

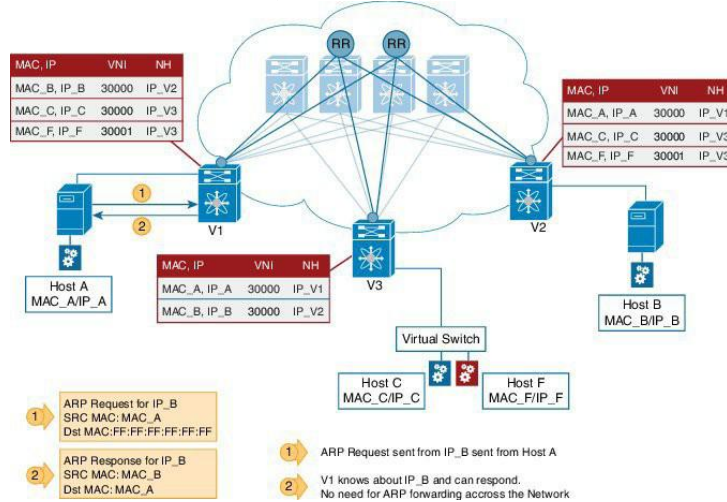
3.1.5 ARP Suppression

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 12 : ARP Suppression



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.

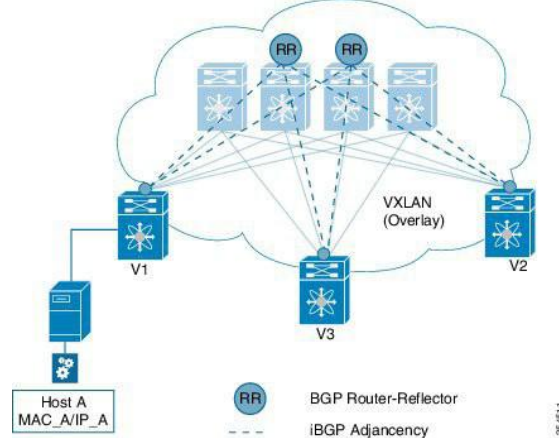
Unknown unicast traffic from an end host is by default flooded in the VLAN. It is possible to avoid the flooding of this traffic to the overlay network without affecting the flooding of this traffic on local host/server ports attached to the ToR switch. Use the `suppress-unknown-unicast` command to do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

3.1.6 Performing End Host Detection, Deletion and Move

End host detection by a VTEP device

Figure 13 : End Host Detection



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

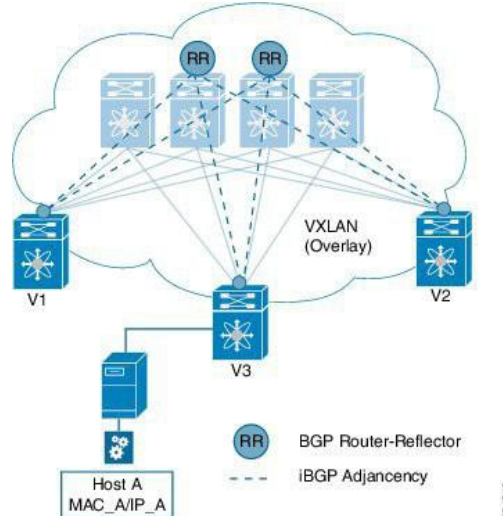
1. VTEP V1 learns Host A's MAC and IP address (MAC_A and IP_A).
2. V1 advertises MAC_A and IP_A to the other VTEPs V2 and V3 through the route reflector.
3. The choice of encapsulation (VXLAN) is also advertised.

A sample depiction of Host A related information:

Figure 14 : Host A - Address Distribution Parameters

MAC, IP	VNI (L2)	VNI (L2)	NH	Encap	Seq
MAC_A, IP_A	30000	50000	IP_V1	3:VXLAN	0

Figure 15 : 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.

Figure 16 : Host A – Updated Parameters

MAC, IP	VNI (L2)	VNI (L2)	NH	Encap	Seq
MAC_A, IP_A	30000	50000	IP_V3	3:VXLAN	1

1. 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.

Pinging from a vPC switch—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.

3.1.7 Guidelines and Limitations for VXLAN BGP EVPN

VXLAN BGP EVPN has the following guidelines and limitations:

- VXLAN EVPN requires the LAN_ENTERPRISE_SERVICES_PKG license.
- Only spine role is supported in Inspur INOS Release 8.4(1).
- VXLAN and Fabric Path features cannot be enabled on the same VDC for F3 and F3 modules.
- VXLAN BGP EVPN is supported on F3 series modules from Inspur INOS Release 8.4(1) onwards.
- Following are not supported for VxLAN BGP EVPN in VDCs having F3 modules:
 - LISP handoff is not supported.
 - Hosts connected behind FEX is not supported.
- Ensure that the root of a Spanning Tree Protocol (STP) domain local to the VXLAN fabric is a VTEP, or placed within the fabric. The STP root should not be outside the VXLAN fabric (below the VTEPs) since it will lead to Layer 2 loops.
 - In VXLAN vPC deployments with F3 modules, the vPC peer-link should be on an isolated ASIC instance from the Layer 3 core ports. After changing the peer-link ports to an isolated ASIC instance, ensure that you reload the switch.
 - In a VXLAN vPC deployment with F3 modules and connected FEX devices, unknown unicast traffic from a remote source to the vPC switch pair may result in duplicate traffic if the MAC address is known to the vPC switches. Both vPC switches will receive a copy of the packet and forward it to the receiver behind FEX.
 - In a VXLAN vPC deployment with F3 modules, known unicast traffic from a remote source to the vPC switch pair may result in loss of traffic if the MAC address is no longer known to the vPC switch pair.
 - In a VXLAN vPC deployment with peer switch, encapsulation profile, and bridge domain configurations, the vPC secondary peer switch does not generate or process BPDUs for bridge domains.
 - If the vPC feature is enabled on a non-vPC (standalone) switch, the NVE source loopback interface will be in shutdown state after an upgrade. To restore the interface to up state, remove the vPC feature using the no feature vpc command in global configuration mode, as shown below.


```
switch(config)# no feature vpc
```

• ARP suppression is only supported for a VNI if the VTEP hosts the First-Hop Gateway (Distributed Anycast Gateway) for this VNI. The VTEP and the SVI for this VLAN have to be properly configured for the distributed anycast gateway operation, for example, global anycast gateway MAC address is configured and anycast gateway feature with the virtual IP address is on the SVI.

The following table lists the VXLAN feature support matrix.

Feature	Inspur INOS Release 8.4(1)	Inspur INOS Release 8.4(1) (F3 Modules)	Inspur INOS Release 8.4(1) (F3 Modules)	Inspur INOS Release 8.4(1) (F3 Modules)	Inspur INOS Release 8.4(1) (F3 Modules)
IPv4/v6 unicast L3GW	Supported	Supported	Supported	Supported	Supported
LISP Hand-off	Supported	Supported	Supported	Not Supported	Not Supported
IPv4/V6 unicast L2GW	Not Supported	Supported	Supported	Not Supported	Supported
vPC Support	Not Supported	Supported	Supported	Not Supported	Supported
IPv4/v6 multicast L2GW	Not Supported	Supported	Supported	Not Supported	Supported
RP on vPC complex	Not Supported	Supported	Supported	Not Supported	Supported
MPLS (L3VPN) Handoff	Not Supported	Supported	Supported	Not Supported	Supported
PIM Bidir underlay	Not Supported	Supported	Supported	Not Supported	Supported
FEX	Not Supported	Supported	Supported	Not Supported	Not Supported
vPC FEX (A-A/S/T)	Not Supported	Supported	Supported	Not Supported	Not Supported
Auto Config	Not Supported	Supported	Supported	Not Supported	Not Supported
ARP Suppression	Not Supported	Supported	Supported	Not Supported	Supported

3.2 Configuring VXLAN BGP EVPN

3.2.1 BGP EVPN and Overlay Configuration

The following BGP, EVPN and overlay configurations are required for the Inspur CN12700 Series switches with F3 modules:

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. (This configuration is applicable only to F3 modules).
3. Layer 3 VNI configurations for a tenant.

Inspur INOS Release 8.4(1) supported only the border spine functionality. Inspur INOS Release 8.4(1) supports the Inspur CN12700 leaf functionality for F3 modules.

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 underlay routing protocol.

(config) #

```
featur
e nv
overla
y
featur
e bgp
featur
e vni
nv overlay evpn
```

Configure the anycast gateway MAC address

(config) #

```
fabric forwarding anycast-gateway-mac
0202.0002.0002
```

Configure BGP L2VPN EVPN address family

(config) #

```
router bgp 100
neighbor 10.1.1.53
remote-as 100
address-family
ipv4 unicast
address-family
l2vpn evpn
send-community
extended
```

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 Inspur
  dot1q 50 vni 30000
```

Associate the encapsulation profile with the server facing interface

(config) #

```
interface Ethernet 1/12
  no
  shud
  own
  no
  switc
  hport
  service instance 1 vni
  encapsulation profile
  Inspur default
  no shutdown
```

Create a loopback interface and assign an IP address to it

(config) #

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

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

(config) #

```
interfa
  ce
  nve
  1 no
  shut
  down
  source-interface
  loopback0 host-
  reachability protocol
  bgp member vni 30000
  mcast-group 224.1.33.3
```

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 12 rd auto
  route-target
  import auto
  route-target
  export auto
```

Note that with the Inspur CN12700 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 address + associated host route (optional)’ of servers attached to the VTEP, for the specified Layer 2 VNI. 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 to the customer VRF
Enable VRF route distinguisher and VRF route target functions for the Layer 3 VNI
(config) #**

```
vrf
  conte
  xt
  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
```

VXLAN BGP EVPN 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 (EVI) 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  201

```

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 encaps: 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 12vpn 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 ?

```

3.3 Feature History for VXLAN BGP EVPN

This table lists the release history for this feature.

Table 2 : Feature History for VXLAN BGP EVPN

Feature Name	Releases	Feature Information
VXLAN BGP EVPN	8.4(1)	This feature was introduced.
	8.4(1)	Support for F3 modules is introduced.

CHAPTER 4 Configuring ACI WAN Interconnect

This chapter contains the following sections:

- VXLAN EVPN - MPLS L3VPN for ACI Fabric.

4.1 VXLAN EVPN - MPLS L3VPN for ACI Fabric

4.1.1 Prerequisites for Configuring ACI WAN Interconnect

- A Inspur CN12700 Series switch with an F3 line card.

4.1.2 Feature History for ACI WAN Interconnect

This table lists the release history for this feature.

Table 3: Feature History for ACI WAN Interconnect

Feature Name	Releases	Feature Information
ACI WAN Interconnect	8.4(1)	This feature was introduced for the F3 line card. Support for MPLS L3VPN as a mechanism or transport outside the ACI fabric was introduced for the F3 line card. Note VRF-Lite and LISP technologies are not supported for this release.
	8.4(1)	Support for the VRF IP routing (or VRF-Lite) was introduced for the F3 line card.
	8.4(1)	Support for VRF-Lite and MPLS L3VPN was introduced for the F3 line card, in addition to the existing support for the F3 line card. Support for LISP as a mechanism or transport outside the ACI fabric was introduced for the F3 line card.
	8.4(1)	Support for VRF-Lite and MPLS L3VPN was introduced for the F3 line card.

4.1.3 Overview of VXLAN EVPN - MPLS L3VPN for ACI Fabric

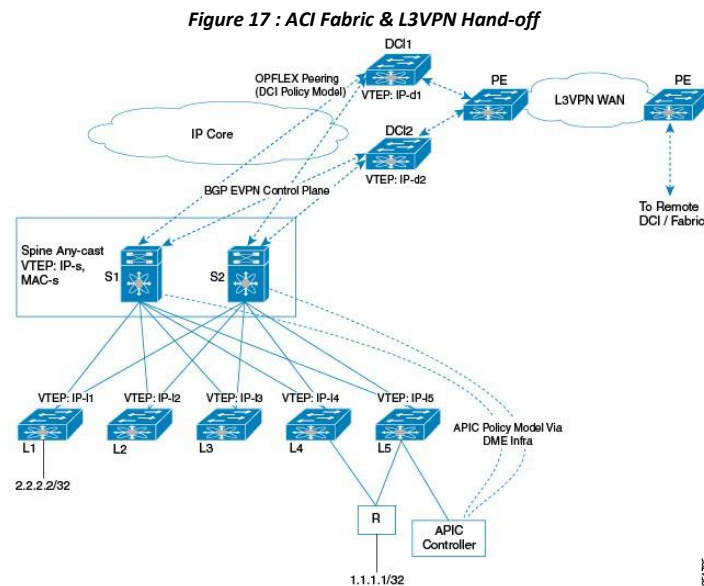
ACI WAN Interconnect is a multi-platform, multi-OS Data Center Interconnect (DCI) architecture. It connects multi-tenant VXLAN data center fabrics over L3VPN.

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

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

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

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



The MPLS L3VPN hand off scenario is explained below.

- BGP-EVPN peering from DCI gateways to two spines in a POD.
- Spines advertise host OR prefix routes for hosts directly behind a leaf, with the Spine Any-cast IP VTEP (IP-s) as the next-hop. These are mostly public BD subnets advertised to outside world.
- Spines can relay a transit route advertised by ACI leaves with leaf VTEPs as next-hops to be used as ECMP paths.
- North-to-South traffic tunneled from DCI to Spine any-cast IP (can land on any of the Spines) or the ECMP is tunneled directly to advertise ACI-leaf VTEPs.
- North-to-South traffic tunneled to Spine will get routed on spine-to-leaf based on /32 lookup.
- Routes advertised from DCI to Spine will get reflected to leaves with the DCI VTEP as the next-hop.
- South-to-North traffic will get routed on the leaves, and ECMP is tunneled directly to the two DCIs.
- Downstream assigned per-VRF VNIDs are advertised by DCI and ACI VTEPs.
- DCI tenant configuration object model are pushed from APIC to Spine to DCI via the OpFlex framework.
- Spines advertise public BD subnet host or prefix routes for hosts directly behind a leaf, with the Spine Any-cast IP VTEP (IP-s) as the next-hop.
- Physical and underlay L3 connectivity between the DCIs and Spines can be via an infrastructure IP network in between or via direct layer 3 sub-interfaces.

4.1.4 Spine – DCI Connectivity

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

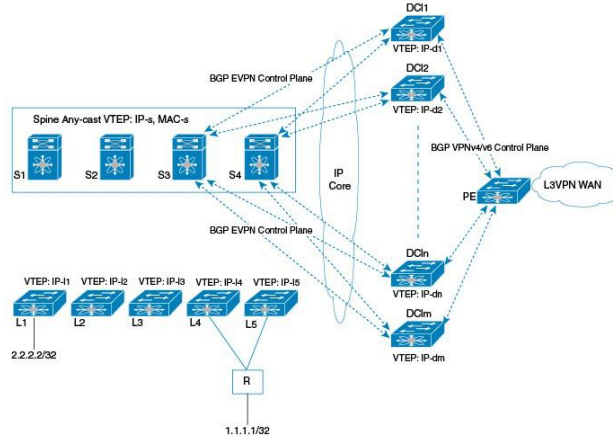
4.1.5 Spine – DCI BGP EVPN Session

BGP session between the Spine and DCI gateway can be eBGP or iBGP. eBGP is the commonly used topology. The MPLS-L3VPN hand-off for ACI fabric can be deployed using one of the following topologies:

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

Single POD With Multiple DCI Gateways

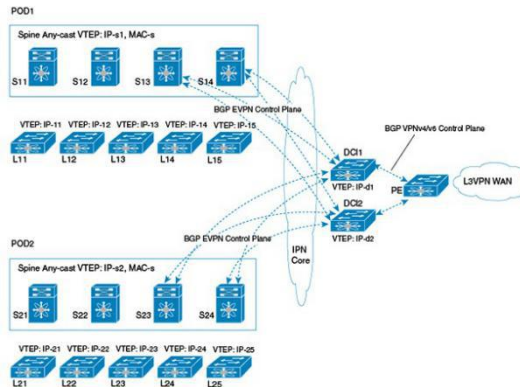
Figure 18 : Single POD with Multiple DCI Gateways



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

Multi-POD With Shared DCI Gateways

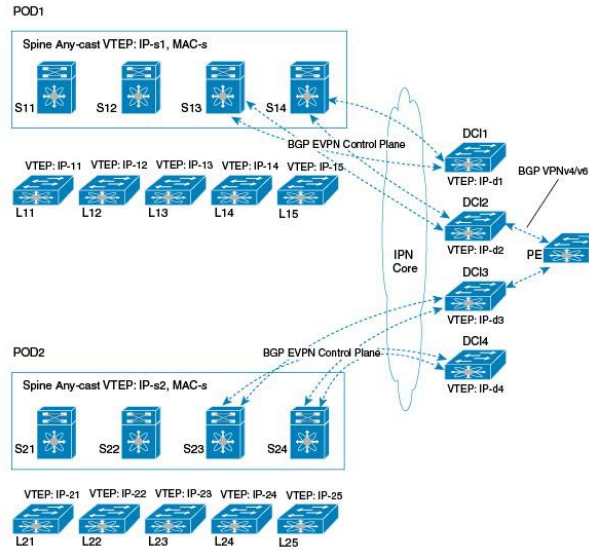
Figure 19 : Multi-POD with Shared DCI Gateway



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

Multi-POD With Separate DCI Gateways

Figure 20 : Multi-POD with Separate DCI Gateways



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

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

4.1.6 OpFlex DCI Auto-Configuration

Inspur OpFlex is a southbound protocol in a software-defined network (SDN) designed to facilitate the communications between the SDN Controller and the infrastructure (switches and routers). The goal is to create a standard that enables policies to be applied across physical and virtual switches/routers in a multi-user environment.

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

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

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

4.1.7 Interconnect Policy Provisioning (IPP)

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

The IPP utilizes OpFlex to push policies from ACI fabric to the DCI gateway. Using these policy attributes,

HMM auto-config is triggered to apply the profile along with the attributes to provision the required fabric-facing configurations.

4.1.8 OpFlex Peering and Multi-POD

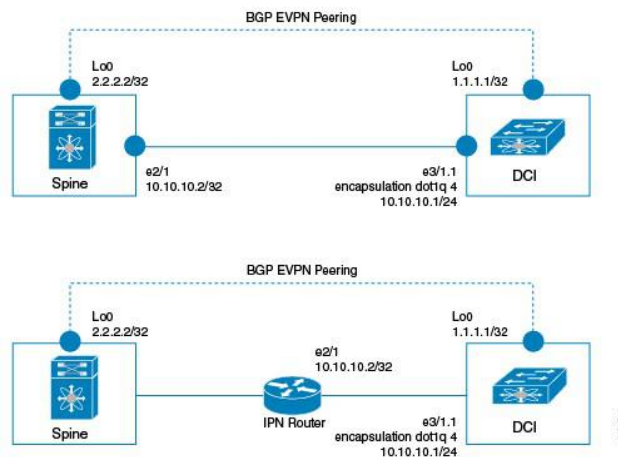
In a multi-POD topology, where the same DCI pair peers with multiple PODs, each POD would be configured as a separate fabric ID and would result in a separate OpFlex framework to be instantiated with the respective spine peers in that fabric that would result in OpFlex sessions to the spine peers within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

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

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

4.1.9 DCI Auto-Configuration Scenario

Figure 21 : DCI Auto-Configuration]



DCI could have the following scenarios:

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

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

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

Spine — DCI (DCI directly connected to multiple spines)

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

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

The following sections describe the OpFlex configuration steps.

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

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

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

#BGP Fabric and WAN Peering

```
router bgp 65000
  address-
    family
      l2vpn evpn
      allow-vni-
        in-ethertag
      ! EVPN Neighbor
      neighbor 2.2.2.1
        remote-as 75000
      address-family
        l2vpn evpn
        import vpn unicast reoriginate
      ! WAN Peering
      neighbor 11.11.11.1 remote-as 65000
      update-source
        loopback 0
      address-family
        vpv4 unicast
import l2vpn evpn reoriginate
```

DCI TEP IP

```
interface loopback0
  ipv4 address 1.1.1.1/32
```

underlay fabric facing L3 interface

```
interface e3/1
  ipv4 address 10.10.10.1/24
```

VXLAN local TEP

```
interface NVE 1
  source-interface
    loopback0 host-
```

```

    reachability
    protocol      bgp
    unknown-peer-
    forwarding    enable
    vxlan        udp    port
    48879

# underlay routing
router ospf area 0
  interfac
  e
  loopback
  0
  interfac
  e e3/1

# DCIs learn reachability to all ACI TEP IPs via OSPF or ISIS
router ospf 100
  router-id 40.0.0.9
  area 0.0.0.100 nssa

# Configuring a peer in downstream (vni) mode
interface nve 1
[no]      vni      assignment
          downstream [all]
          Peer-ip <ip address
          1>
          Peer-ip <ip address 2>
          .....
          Peer-ip <ip address 3>

# Changing the default forwarding behavior

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

```

2) Configuring profile templates

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

```

# MPLS L3VPN hand-off common profile
configure profile vrf-common-mpls-
  l3vpn-dc-edge vrf context $vrfName
  vni
  $include_vrfSegme
  ntId rd auto
  address-family ipv4 unicast
  route-target
  $include_client_import_ipv4_bgpRT_1 evpn route-
  target export $include_client_export_ipv4_bgpRT_1
  evpn route-target import
  $include_client_import_ipv4_bgpRT_2 evpn route-
  target export $include_client_export_ipv4_bgpRT_2
  evpn route-target import
  $include_client_import_ipv4_bgpRT_3 evpn route-
  target export $include_client_export_ipv4_bgpRT_3
  evpn route-target import
  $include_client_import_ipv4_bgpRT_4 evpn route-
  target export $include_client_export_ipv4_bgpRT_4
  evpn route-target import
  $include_client_import_ipv4_bgpRT_5 evpn route-
  target export $include_client_export_ipv4_bgpRT_5
  evpn route-target import
  $include_client_import_ipv4_bgpRT_6 evpn route-
  target export $include_client_export_ipv4_bgpRT_6

```

```

    evpn                route-target                import
    $include_client_import_ipv4_bgpRT_7  evpn route-
target    export    $include_client_export_ipv4_bgpRT_7
    evpn                route-target                import
    $include_client_import_ipv4_bgpRT_8  evpn route-
target    export    $include_client_export_ipv4_bgpRT_8
    evpn
address-family ipv6 unicast
    route-target                import
    $include_client_import_ipv6_bgpRT_1  evpn route-target
    export    $include_client_export_ipv6_bgpRT_1  evpn
    route-target                import
    $include_client_import_ipv6_bgpRT_2  evpn route-target
    export    $include_client_export_ipv6_bgpRT_2  evpn
    route-target                import
    $include_client_import_ipv6_bgpRT_3  evpn route-target
    export    $include_client_export_ipv6_bgpRT_3  evpn
    route-target                import
    $include_client_import_ipv6_bgpRT_4  evpn route-target
    export    $include_client_export_ipv6_bgpRT_4  evpn
    route-target                import
    $include_client_import_ipv6_bgpRT_5  evpn route-target
    export    $include_client_export_ipv6_bgpRT_5  evpn
    route-target                import
    $include_client_import_ipv6_bgpRT_6  evpn route-target
    export    $include_client_export_ipv6_bgpRT_6  evpn
    route-target                import
    $include_client_import_ipv6_bgpRT_7  evpn route-target
    export    $include_client_export_ipv6_bgpRT_7  evpn
    route-target                import
    $include_client_import_ipv6_bgpRT_8  evpn route-target
    export $include_client_export_ipv6_bgpRT_8 evpn*
router bgp $asn vrf $vrfName
    address-family ipv4
        unicast advertise
        l2vpn evpn
        label-allocation-mode
    per-vrf address-family
    ipv6 unicast
    advertise    l2vpn
    evpn
    label-allocation-mode
per-vrf interface nve
    $nveId
    member vni $include_vrfSegmentId
associate-vrf configure terminal

```

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

```

# Enable IPP feature
feature ipp

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

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

```



```

system fabric bridge-domain 2000-3000

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

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

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

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

# Add IPP owned per ACI/OpFlex instance configuration
ipp
fabric 1
opflex-peer
10.2.2.2 8009 ssl
encrypted
fabric 2
opflex-peer
10.2.3.2 8009 ssl
encrypted

```

4.1.10 Show and Debug Command Examples

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

```

switch# show ipp internal ?
event-history Show various event
logs of IPP mem-stats Dynamic
memory stats
pss Internal IPP
pss info work-info Internal IPP
worker thread info

switch# show ipp internal debug
IPP Debug information
Debug Flags : Off
Debug-filters : Off

switch# show ipp internal
event-history ? errors Show

```

```

error logs of IPP events
      Show IPP process
events
ha      Show IPP process HA events
msgs   Show various message
logs of IPP opflex      Show IPP
process opflex events periodic
Show IPP process periodic
events trace Show processing
logs of IPP

```

```

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

```

```

switch# show ipp internal event-history events
Process Event logs of IPP
2016 May 27 17:08:16.457294 ipp [6452]: [6495]: Unconfig ipp tenant
dci_coke-99
218 command
2016 May 27 17:08:15.105985 ipp [6452]: [6495]: Unconfig ipp tenant
dci_coke-98
235 command
2016 May 27 17:08:14.370121 ipp [6452]: [6495]: Unconfig ipp tenant
dci_coke-97
216 command
2016 May 27 17:08:13.133061 ipp [6452]: [6495]: Unconfig ipp tenant
dci_coke-96
198 command
2016 May 27 17:08:12.331485 ipp [6452]: [6495]: Unconfig ipp tenant
dci_coke-95
201 command
2016 May 27 17:08:11.052111 ipp [6452]: [6495]: Unconfig ipp tenant
dci_coke-94
209 command

2016 May 27 17:08:10.341556 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-93
173 command
2016 May 27 17:08:09.061573 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-92
184 command
2016 May 27 17:08:08.376121 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-91

```

```

255 command
2016 May 27 17:08:07.183026 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-90
260 command
2016 May 27 17:08:06.483588 ipp [6452]: [6495]: Unconfig ipp tenant dci_coke-9 2
.....

switch# show ipp internal event-history msgs
Msg events for IPP Process
1) Event:E_DEBUG, length:45, at 581444 usecs after Mon May 30
   11:34:23 2016 [100] [19461]: nvdb: transient thread created

2) Event:E_DEBUG, length:82, at 579664 usecs after Mon May 30 11:34:23
   2016 [100] [6495]: comp-mts-rx opc - from sap 19164 cmd
   ipp_show_internal_event_h
   ist_cmd

3) Event:E_DEBUG, length:49, at 882139 usecs after Mon May 30
   11:33:45 2016 [100] [19410]: nvdb: terminate transaction failed
.....

switch# show ipp internal event-history opflex
Process opflex logs of IPP
2016 May 30 11:05:46.967301 ipp [6452]: [4710]:
Processor.cpp:261:resolveObj(): Resolving policy
/PolicyUniverse/PolicySpace/coke-72/GbpRoutingDomain/coke-72/ 2016 May
30 11:05:46.967242 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-
63/GbpRoutingDomain/coke-63/ 2016 May 30 11:05:46.967165 ipp [6452]:
[4710]: Processor.cpp:261:resolveObj(): Resolving policy
/PolicyUniverse/PolicySpace/coke-28/GbpRoutingDomain/coke-28/ 2016 May
30 11:05:46.716185 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-
45/GbpRoutingDomain/coke-45/ 2016 May 30 11:05:46.715810 ipp [6452]:
[4710]: Processor.cpp:261:resolveObj(): Resolving policy
/PolicyUniverse/PolicySpace/coke-19/GbpRoutingDomain/coke-19/ 2016 May
30 11:05:46.715754 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-
90/GbpRoutingDomain/coke-90/ 2016 May 30 11:05:46.715696 ipp [6452]:
[4710]: Processor.cpp:261:resolveObj(): Resolving policy
/PolicyUniverse/PolicySpace/coke-73/GbpRoutingDomain/coke-73/ 2016 May
30 11:05:46.715639 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-
82/GbpRoutingDomain/coke-82/ 2016 May 30 11:05:46.715580 ipp [6452]:
[4710]: Processor.cpp:261:resolveObj(): Resolving policy
/PolicyUniverse/PolicySpace/coke-55/GbpRoutingDomain/coke-55/ 2016 May
30 11:05:46.715214 ipp [6452]: [4710]: Processor.cpp:261:resolveObj():
Resolving policy /PolicyUniverse/PolicySpace/coke-
29/GbpRoutingDomain/coke-29/ 2016 May 30 11:05:46.715153 ipp [6452]:
[4710]: Processor.cpp:261:resolveObj():
.....

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

```

```
2016 May 27 17:05:28.351387 ipp [6452]: [6493]: IPP Worker thread in progress
2016 May 27 17:05:28.347969 ipp [6452]: [6493]: IPP Worker thread in progress
```

```
2016 May 27 17:05:28.343087 ipp [6452]: [6493]: IPP Worker thread in progress
.....
```

```
switch# show ipp internal event-history trace
```

```
Trace logs of IPP
```

```
2016 May 30 01:07:32.962911 ipp [6452]: [6492]: sysmgr consumed
mts[2558] message MTS_OPC_SYSMGR_PSS_SHRINK, drop
2016 May 30 01:07:32.962893 ipp [6452]: [6492]: sysmgr processing
mts[2558] message MTS_OPC_SYSMGR_PSS_SHRINK
2016 May 28 01:07:29.561840 ipp [6452]: [6492]: sysmgr consumed
mts[2558] message MTS_OPC_SYSMGR_PSS_SHRINK, drop
2016 May 28 01:07:29.561818 ipp [6452]: [6492]: sysmgr processing
mts[2558] message MTS_OPC_SYSMGR_PSS_SHRINK
2016 May 27 17:06:50.843023 ipp [6452]: [6492]: Done processing
MTS[250880] message MTS_OPC_HMM dropped
.....
```

```
switch# show ipp internal mem-stats
```

```
Mem stats for IPP Process
```

```
Num blocks      User size      Total          size
                Library
```

```
-----
---
304             155244         162128         ipp
332             64567          70336         ld-2.15.so
120             5712           8504          libc-2.15.so
3              60             120           libdl-2.15.so
1              8              24            librt-2.15.so
22877          1053813        1533896       libstdc++.so.6.0.16
28             2294           2872          libvlan_mgr.so.0.0.0
12             2860           3144          libsvifdb.so.0.0.0
1              408            432           libltlmap.so.0.0.0
9              576            792           libindxobj.so.0.0.0
230            6472           11072         libast_db.so.0.0.0
116            2304           4648          libavl.so.0.0.0
4              8256           8352          libxos_lpss.so.0.0.0
14             2604           2928          libfsrv.so.0.0.0
3544           148784         193384        libcrypto.so.1.0.0
99             6702           8728          libsd.so.0.0.0
358            11340          18544         libglib-2.0.so.0.3200.3
2              80             128           libstartup-config.so.0.0.0
32             151948         152648        libsdwrap.so.0.0.0
162            5952           9296          libfsrv_client.so.0.0.0
1              256            280           libcmd.so.0.0.0
94             6195           8280          libcli_common.so.0.0.0
28             31332          31960         librs.so.0.0.0
5              41120          41232         libtcp_clt.so.0.0.0
3              152            216           libutils.so.0.0.0
88             24520          26680         libpss.so.0.0.0
3              528            600           libmts.so.0.0.0
9              1800           2016          libsysmgr.so.0.0.0
2              65804          65848         libopflex.so.0.0.0
2              512            560           libuv.so.0.0.0
689            34556          51008         libmtrack.so.0.0.0
```

```
Grand totals:
```

```
29172 Blocks
```

1836759 User Bytes
2420656 Total Bytes

switch# **show ipp internal pss**

PSS Data

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

.....
switch# **show ipp internal vrf-db**
1: Vrf: dci_coke-1
0: RT v4:(1:1,1:1) v6:(1:1,1:1)

```

Host Id: 289, # tenants: 1
      Tenant Id: 289
2: Vrf: dci_coke-10
   0: RT v4:(1:10,1:10) v6:(1:10,1:10)
      Host Id: 266, # tenants: 1
      Tenant Id: 266
3: Vrf: dci_coke-100
   0: RT v4:(1:100,1:100) v6:(1:100,1:100)
      Host Id: 346, # tenants: 1
      Tenant Id: 346
4: Vrf: dci_coke-11
   0: RT v4:(1:11,1:11) v6:(1:11,1:11)
      Host Id: 351, # tenants: 1
      Tenant Id: 351
5: Vrf: dci_coke-12
   0: RT v4:(1:12,1:12) v6:(1:12,1:12)
      Host Id: 281, # tenants: 1
      Tenant Id: 281
6: Vrf: dci_coke-13
   0: RT v4:(1:13,1:13) v6:(1:13,1:13)
      Host Id: 275, # tenants: 1
      Tenant Id: 275
7: Vrf: dci_coke-14
   0: RT v4:(1:14,1:14) v6:(1:14,1:14)
      Host Id: 305, # tenants: 1
      Tenant Id: 305

```

```

.....
switch# show ipp internal work-info

```

```

IPP Worker information

```

```

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

```

```

switch# show ipp fabric

```

```

Global info:

```

```

config-profile
defaultNetworkMplsL3vpnDcProfile
include-config-profile vrf-common-mpls-
l3vpn-dc-edge local-vtep nve 1
bgp-as 100
identity 1.1.1.1

```

```

Fabric 1 (Healthy)

```

```

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

```

```

Tenant Policies

```

```

1: Fabric Vrf: coke-1:coke-1, Vrf:
dci_coke-1 RT v4:(1:1,1:1)
v6:(1:1,1:1)
Id 289, HostId: 289 flags 0x0
2: Fabric Vrf: coke-10:coke-10, Vrf:
dci_coke-10 RT v4:(1:10,1:10)
v6:(1:10,1:10)
Id 266, HostId: 266 flags 0x0

```

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

!Command: show running-config ipp
!Time: Wed Jun  1 08:37:18 2016

version 7.3(0)DX(1) feature ipp

ipp
  profile-map profile defaultNetworkMplsL3vpnDcProfile include-
  profile vrf-commo n-mpls-l3vpn-dc-edge
  local-vtep nve 1
  bgp-as 100

  identity 1.1.1.1
  fabric 1
  opflex-peer
  20.4.11.1 8009 ssl
  encrypted
ipp tenant
dci_coke-1 289 ipp
tenant dci_coke-10
266 ipp tenant
dci_coke-100 346
ipp tenant
dci_coke-11 351 ipp
tenant dci_coke-12
281 ipp tenant
dci_coke-13 275 ipp
tenant dci_coke-14
305 ipp tenant
dci_coke-15 292 ipp
tenant dci_coke-16
339 ipp tenant
dci_coke-17 323 ipp
tenant dci_coke-18
330 ipp tenant
dci_coke-19 361 ipp
tenant dci_coke-2
310 ipp tenant
dci_coke-20 350 ipp
tenant dci_coke-21
283 ipp tenant
dci_coke-22 272 ipp
tenant dci_coke-23
299 ipp tenant
dci_coke-24 294 ipp
tenant dci_coke-25
337 ipp tenant
dci_coke-26 326 ipp
tenant dci_coke-27
334 ipp tenant
dci_coke-28 363 ipp
tenant dci_coke-29
356 ipp tenant
dci_coke-3 343 ipp
tenant dci_coke-30
277 ipp tenant
```

```
dci_coke-31 307 ipp
tenant dci_coke-32
293 ipp tenant
dci_coke-33 341 ipp
tenant dci_coke-34
321 ipp tenant
dci_coke-35 329 ipp
tenant dci_coke-36
269 ipp tenant
dci_coke-37 352 ipp
tenant dci_coke-38
285
.....
switch# debug ipp ?
all          All debugs
cli          CLI command processing debugs
event       IPP events
ha          HA
related     debugs  hmm
            IPP HMM api
debug opflex IPP
opflex      debugs
periodic    IPP  events
periodic
```


CHAPTER 5 Campus Fabric

This chapter contains the following sections:

- Campus Fabric.
- Feature History for Campus Fabric.

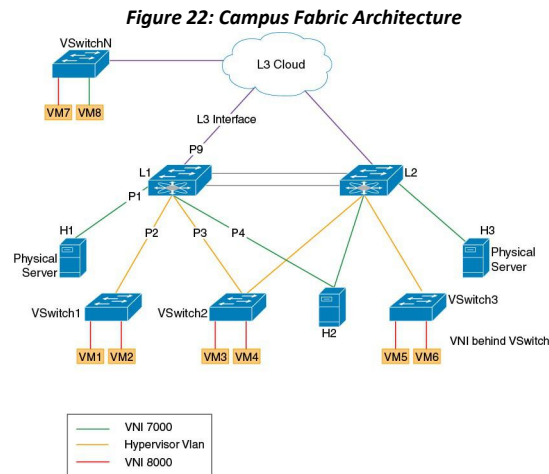
5.1 Campus Fabric

5.1.1 Overview of Campus Fabric

The Campus Fabric feature provides the basic infrastructure for building virtual networks based on policy-based segmentation constructs. Fabric overlay provides services such as host mobility and enhanced security, which are in addition to normal switching and routing capabilities.

This feature enables a LISP-based Control Plane for VXLAN Fabric. The Inspur CN 12700 Series with F3 Module acts as a fabric border which connects traditional Layer 3 networks or different fabric domains to the local fabric domain, and translates reachability and policy information from one domain to another. However inter-fabric connectivity on a same switch is not supported.

Inspur CN12700 is positioned as a fabric border node in the Campus Fabric architecture.



The key elements of the Campus fabric architecture are explained below.

Campus Fabric : The Campus Fabric is an instance of a "Network Fabric". A Network Fabric describes a network topology where data traffic is passed through interconnecting switches, while providing the abstraction of a single Layer-2 and/or Layer-3 device. This provides seamless connectivity, independent of physical topology, with policy application and enforcement at the edges of the fabric. Enterprise fabric uses IP overlay, which makes the network appear like a single virtual router/switch without the use of clustering technologies. This logical view is independent of the control plane used to distribute information to the distributed routers or switches.

Fabric Edge Node : Fabric edge nodes are responsible for admitting, encapsulating/decapsulating and forwarding traffic to and from endpoints connected to the fabric edge. Fabric edge nodes lie at the perimeter of the fabric and are the first points for attachment of the policy. It is to be noted that the endpoints need not be directly attached to the fabric edge node. They could be indirectly attached to a fabric edge node via an intermediate Layer-2 network that lies outside the fabric domain.

Traditional Layer-2 networks, wireless access points or end-hosts are connected to Fabric Edge nodes.

Fabric Intermediate Node: Fabric intermediate nodes provide the Layer-3 underlay transport service to fabric traffic. These nodes are pure layer-3 forwarders that connect the Fabric Edge and Fabric Border nodes.

In addition, Fabric intermediate nodes may be capable of inspecting the fabric metadata and could apply policies based on the fabric metadata (not mandatory). However, typically, all policy enforcement is at the Fabric Edge and Border nodes.

Fabric Border Node : Fabric border nodes connect traditional Layer-3 networks or different fabric domains to the Campus Fabric domain.

If there are multiple Fabric domains, the Fabric border nodes connect a fabric domain to one or more fabric domains, which could be of the same or different type. Fabric border nodes are responsible for translation of context from one fabric domain to another. When the encapsulation is the same across different fabric domains, the translation of fabric context is generally 1:1. The Fabric Border Node is also the device where the fabric control planes of two domains exchange reachability and policy information.

APIC-EM Controller : This is the SDN controller developed by the Enterprise Networking Group. This controller serves both Brownfield and Greenfield deployments. Campus Fabric service will be developed on the APIC-EM controller. This service will drive the management and orchestration of the Campus Fabric, as well as the provision of policies for attached users and devices.

Fabric Header: Fabric header is the VXLAN header which carries the segment ID(VNI) and user group(SGT). SGT is encoded in the reserved bits of the VXLAN header.

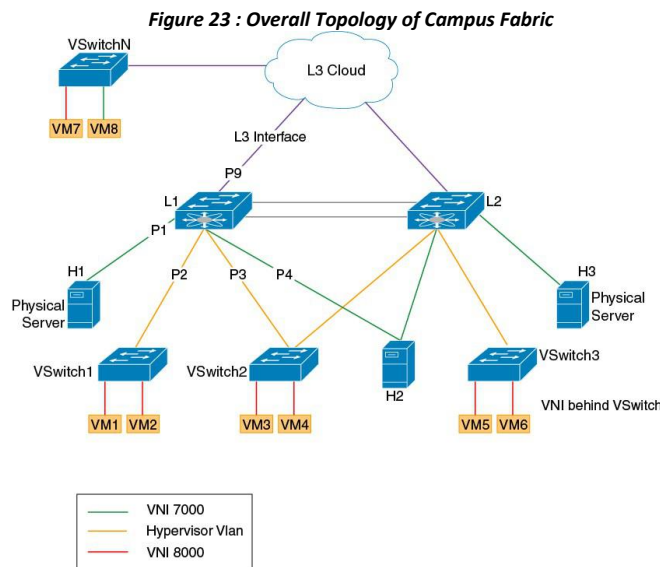
Inspur S6650 is positioned as the fabric edge and Inspur CN12700 is positioned as the fabric border in this architecture. LISP is the control plane in the campus fabric architecture and it programs the VXLAN routes. LISP is enhanced to support VXLAN routes for Campus Fabric architecture.

The following features are supported on Inspur CN12700:

- LISP control plane pushing VXLAN routes
- VXLAN L3GW (VRF-lite Hand-off)
- Optimal Tenant L3 Multicast (ASM/Bidir/SSM) based on LISP Multicast (Ingress Replication over unicast core)
- IS-IS as underlay
- TTL propagation

5.1.2 VXLAN Encapsulation for Layer-3 LISP Configuration

This section summarizes only the steps that are used for configuring LISP for a hand-off from VXLAN on the border spine or border leaf switch.



35-4017

```
feature-set fabric hostname PxTR1

feature telnet feature bgp feature pim
feature isis

feature lisp

feature interface-vlan system bridge-domain
100 feature nv overlay feature vni
vni 5000

vlan 1
bridge-domain 100

route-map LISP-
RMAP permit 10
bridge-domain 100

/* UNDERLAY VRF*/
vrf context
  core
  description
  "UNDERLAY VRF"
  ipv6 lisp itr-
  etr
  ip lisp itr-etr
  ipv6 lisp itr map-
  resolver 9.9.9.9 ip
  lisp itr map-resolver
  9.9.9.9
  ip lisp etr map-server 9.9.9.9 key 3
  a97b0defe7b8ff70 ip lisp multicast
  lisp encapsulation vxlan

/* OVERLAY VRF */
vrf context
  vrf5000
  description
  "OVERLAY VRF "
  vni 5000
  ip pim rp-address 200.1.2.1 group-list
  225.0.0.0/24 ip lisp proxy-itr 10.1.1.1
ip lisp proxy-etr
  lisp instance-
  id 5000
  ip lisp locator-vrf core
  ip lisp map-cache 100.0.1.0/24
  map-request ip lisp multicast
  lisp
  encapsulation
  vxlan address-
  family ipv4
  unicast
  route-target import 3:3
  route-target export 1:1

bridge-domain 100
  member vni 5000

interface Bdi100
  description "BDI in
  OVERLAY vrf" no
  shutdown
  vrf member vrf5000 no
  ip redirects ip
  forward
```

```
ip pim sparse-mode

interface nve1 no shutdown
  source-interface
  loopback0      host-
  reachability
  protocol        lisp
  member vni      5000
  associate-vrf

interface Ethernet1/5
  description PxTR1 to SPINE1
  link(UNDERLAY VRF) vrf member core
  ip address
  10.1.1.1/24 isis
  circuit-type
  level-1-2 ip
  router isis 100
  ip pim sparse-mode
  no shutdown

interface
  Ethernet5/1
  no shutdown

interface Ethernet5/1.1
  description PxTR1 to CORE vrf
  vrf5000(OVERLAY VRF) encapsulation dot1q 2
  vrf member vrf5000
  ip address
  80.1.1.1/24 ip
  pim sparse-
  mode
  no shutdown

interface loopback0
  description "Source Locator
  loopback" vrf member core
  ip address 1.1.1.1/32
  isis circuit-type
  level-1-2 ip router
  isis 100
  ip pim sparse-mode

interface loopback100
  Description "OVERLAY VRF
  loopback" vrf member
  vrf5000
  ip address
  111.1.1.1/32 ip
  pim sparse-mode

/* IGP on the UNDERLAY VRF
*/ router isis 100
  net 49.0001.1111.1111.1111.00
  vrf core
  net 49.0001.1111.1111.1111.00
  vrf vrf5000

/* BGP neighbor towards the
CORE */ router bgp 100
  description "
```

```

router-id
12.12.12.13
vrf vrf5000
  address-family ipv4 unicast
    redistribute lisp route-map
    LISP-RMAP
    redistribute direct route-map
  LISP-RMAP neighbor 80.1.1.2
  remote-as 100
  Description "BGP neighbor to the CORE
  Router" address-family ipv4 unicast
  address-family ipv6 unicast

```

SGT Propagation, Termination, and Generation

The security group tag (SGT) allows the network to enforce the access control policy by enabling the endpoint device to act upon the SGT to filter traffic.

At the ingress point, traffic from the source is tagged with an SGT containing the security group number of the source entity. The SGT is propagated with the traffic across the domain. At the egress point, an egress device uses the source SGT and the security group number of the destination entity to determine which access policy to apply from the security group access control lists (SGACL) policy matrix.

The least significant 16 bits in the reserved field of the VXLAN header is used to carry the SGT information.

For traffic ingressing the site from internet a mechanism is needed to classify the packets as internet packets and drive SGT based on the classification. This SGT is used in the reserved field of the VXLAN header during VXLAN encapsulation.

For traffic egressing the site the SGT field should be used from the reserved field during VXLAN decapsulation and policy enforcement can be done based on the sg tag. This is where F3 module acts as a PETR. This is enabled using their lisp sgt command.

Multicast Head-end Replication

Head-end replication accomplishes the need of a multicast transport for Overlay Transport Virtualization (OTV) control plane communications. Multicast transport is used to let a single OTV update or packets to reach all other OTV devices using a specific multicast group address across domains.

LISP Multicast configuration on an ETR or ITR is covered in the "VXLAN Encapsulation for Layer-3 LISP Configuration" section described above.

TTL Propagation

TTL (Time-to-Live) is a setting for each DNS record that specifies how long a resolver should cache the DNS query before the query expires and a new query needs to be made.

TTL propagation from the inner header to the outer header during VXLAN encapsulation is done based on a CLI. On enabling this CLI, the TTL propagation will be disabled from the inner header to the outer header during encapsulation. This is enabled using the **lisp disable-ttl-propagate** command.

5.2 Feature History for Campus Fabric

This table lists the release history for this feature.

Table 4 : Feature History for Campus Fabric

Feature Name	Releases	Feature Information	
Campus Fabric	8.4(1)	This feature was introduced.	

CHAPTER 6 Campus Fabric Interconnect - MPLS L3VPN

This feature explains a sample Software-Defined Access (SD-Access) network topology comprising two Locator/ID Separation Protocol (LISP) control plane-Virtual eXtensible Local Area Network (VXLAN) data plane based campus fabrics connected through Multiprotocol Label Switching (MPLS) L3VPN. The focus of the feature is the role of the Inspur CN12700 Series border leaf switch which sends end host traffic from the fabric to an end host in a remote fabric over MPLS (through the MPLS core).

This chapter contains the following sections:

- Prerequisites of Campus Fabric Interconnect—MPLS L3VPN.
- Information About Campus Fabric Interconnect—MPLS L3VPN .
- How to Configure Campus Fabric Interconnect—MPLS L3VPN.
- Verifying Campus Fabric Interconnect—MPLS L3VPN.
- Feature History for Campus Fabric Interconnect—MPLS L3VPN.

6.1 Prerequisites of Campus Fabric Interconnect—MPLS L3VPN

- Conceptual and configuration knowledge about VXLAN-with-LISP campus fabrics, since the focus of this feature is the fabric interconnect function.
- Functioning campus fabrics wherein the LISP, VXLAN and other required configurations are enabled. See the "Campus Fabric" chapter for more information.

6.2 Information About Campus Fabric Interconnect—MPLS L3VPN .

6.2.1 Campus Fabric Architecture—Fabric 1

End hosts are attached to Inspur Catalyst switches xTR 1 and xTR 2 which perform the role of LISP xTRs. The LISP control plane extends from the xTRs to PxTR 1 and PxTR 2. Spine1 and Spine 2 are Layer-3 switches used for routing in the underlay, through an interior gateway protocol (IGP) such as Open Shortest Path First (OSPF). Spine 1 is connected to the Map-Server/Map-Resolver (MSMR).

For the overlay, VXLAN is implemented on the xTRs and the PxTRs, and they also perform the role of VXLAN Virtual Tunnel End Points (VTEPs).

The LISP (control plane) and the VXLAN (data plane) overlays begin and terminate between the xTRs and PxTRs.

6.2.2 Traffic Flow Between Fabrics—Campus Fabric Interconnect

PxTR1 and PxTR2 perform the provider edge (PE) function, and are connected to the provider switch P1 in the MPLS/IP core. MPLS L3VPN is implemented on the PxTRs for traffic flow across the fabrics. If Host 1 in Fabric 1 sends traffic to Host 3 in Fabric 2, then this is a sample flow:

- Traffic from Host1 reaches a PxTR, the fabric border switch, since the destination end host is located in a remote site. The PxTR VXLAN decapsulates the packets and sends it towards P1 through MPLS.
- P1 sends the traffic through the MPLS/IP core to the Provider switch P2 which is connected to the fabric border switch of Fabric 2, PxTR 3. P2 forwards the MPLS traffic to PxTR3.
- PxTR3 receives the traffic, removes the MPLS label, and does appropriate lookups as regards to the destination end host. Then, PxTR3 VXLAN encapsulates the packets towards xTR3, since Host 3 is attached to it.
- xTR 3 receives the traffic, VXLAN decapsulates the packets and sends the original packets (sent by Host 1) to Host 3.

6.3 How to Configure Campus Fabric Interconnect—MPLS L3VPN.

6.3.1 Feature Set Configuration

Configure MPLS L3VPN, BGP, LDP, LISP and VXLAN features:

```
PxTR 1(config)# feature-set mpls
feature-set fabric feature bgp feature lisp
feature mpls l3vpn feature mpls ldp feature
nv overlay feature vni
```

• Some configurations, such as LISP and VXLAN features, are already enabled for campus fabric configuration. They are noted here for completeness.

6.3.2 Campus Fabric Configuration

Step 1 Configure VXLAN related commands

Create a bridge domain and associate the corresponding Layer 3 VNI:

```
PxTR 1(config)# vni 6000
system bridge-domain 300
bridge-domain 300
member vni 6000
```

Add the Layer 3 virtual routing and forwarding (VRF) VNI to the VXLAN overlay network and enable LISP reachability:

```
PxTR 1(config)# interface nve1
no shutdown
host-reachability
protocol lisp
source-interface
loopback0 member
vni 6000 associate-
vrf
```

Step 2 Configure LISP related commands

Configure LISP parameters and route distinguisher and route target functions for the *vrf6000* VRF:

```
PxTR 1(config)# vrf context vrf6000
vni 6000
ip lisp proxy-itr
192.0.2.1 ip lisp
proxy-etr
lisp instance-
id 6000 ip lisp
locator-vrf fab0
ip lisp map-cache 198.51.0.0/16
map-request lisp encapsulation
vxlan
rd 6000:6000
address-family ipv4
unicast route-target
import 6000:6000
route-target export 6000:6000
```

The **ip lisp map-cache** command creates a static map-cache entry for reachability to remote Endpoint Identifiers

(EIDs).

Step 3 Configure fabric facing BDI Associate a BDI to the vrf6000 VRF:

```
PxTR 1(config)# interface Bdi300
no shutdown
vrf member
vrf6000 no
ip
redirects
ip forward
```

6.3.3 Campus Fabric Interconnect Configuration

Step 1 Configure MPLS commands on the WAN facing interface:

```
PxTR 1(config)# interface Ethernet1/35.1
mpls ip
description
connect_P1_mpls
encapsulation dot1q
162
ip address 203.0.113.1/30
ip router ospf 299 area
0.0.0.0 no shutdown
```

- After enabling the corresponding interface on P1, an MPLS link is established between PxTR 1 (the PE switch) and P1 (the Provider switch) .

```
PxTR 1(config)# mpls ldp configuration
router-id Lo299 force
```

- The configurations enable the specified loopback interface' s IP address as the Label Distribution Protocol (LDP) router ID.

Step 2 Configure BGP for traffic flow between the fabric border (PE) switch PxTR 1 and the Provider switch P1:

```
PxTR 1(config)# router bgp 100
router-id 209.165.201.1
address-family ipv4 unicast
```

- The IPv4 address family and router ID configurations are enabled.

```
neighbor 209.165.200.225
remote-as 5000
update-source
loopback299 ebgp-
multihop 10
address-family
vpn4 unicast
send-
community
extended exit
address-family ipv4 unicast
```

- BGP neighbor/peer VPNv4 and IPv4 address family configurations are enabled.

```
vrf vrf6000
address-family ipv4 unicast
redistribute lisp route-
map LISP-RMAP
```



```

aggregate-address 198.51.0.0/16
summary-only label-allocation-
mode per-vrf

```

- The **redistribute lisp route-map** command redistributes the LISP map-cache routes into **MP-BGP**.
- Aggregate routes within the vrf6000 VRF are enabled to be distributed to the BGP neighbor.

6.4 Verifying Campus Fabric Interconnect—MPLS L3VPN

You can verify MPLS configurations on a fabric border switch with these verification commands:

6.4.1 Verifying MPLS LDP Configuration

In the following example, you can verify MPLS LDP configuration:

```

PxTR1# show mpls ldp discovery

Local LDP Identifier:
209.165.201.1:0
Discovery Sources:
Interfaces:
Ethernet2/20.1 (ldp):
xmit/recv LDP Id:
203.0.113.1:0

```

6.4.2. Verifying MPLS LDP Neighbor Configuration

In the following example, you can verify MPLS LDP neighbor configuration:

```

PxTR1# show mpls ldp neighbor

Peer LDP Ident: 203.0.113.1:0; Local LDP Ident
209.165.201.1:0 TCP connection: 203.0.113.1.646 -
209.165.201.1.63118
State: Oper; Msgs sent/rcvd: 69/71;
Downstream Up time: 00:53:49
LDP discovery sources:
Ethernet2/20.1, Src IP addr:
192.0.2.250 Addresses bound to
peer LDP Ident:
203.0.113.1 172.16.0.1 192.0.2.250 203.0.113.10

```

6.4.3 Verifying MPLS Label Switching VRF Information

In the following example, you can verify MPLS label switching VRF information:

```

PxTR1# show mpls switching vrf vrf6000

Legend:
(P)=Protected, (F)=FRR active, (*)=more labels in stack.

In-Label VRF
IPv4
Aggregate
Labels 31
vrf
f6000

```

6.5 Feature History for Campus Fabric Interconnect—MPLS L3VPN

This table lists the release history for this feature.

Table 5 : Feature History for Campus Fabric Interconnect—MPLS L3VPN

Feature Name	Release	Feature Information	
Feature History for Campus Fabric Interconnect—MPLS L3VPN	8.4(1)	This feature was introduced. This feature explains how to enable traffic flow across two campus fabrics through MPLS L3VPN. No new commands were introduced for this feature.	

CHAPTER 7 LISP Support for Disjointed RLOC Domains

This chapter contains the following sections:

- LISP Support for Disjointed RLOC Domains.

7.1 LISP Support for Disjointed RLOC Domains

7.1.1 Overview of LISP Support for Disjointed RLOC Domains

Locator/ID Separation Protocol (LISP) implements a level of indirection that enables a new IP routing architecture. LISP separates IP addresses into two address spaces, Endpoint Identifiers (EIDs), which are assigned to end hosts, and Routing Locators (RLOCs), which are assigned to devices that make up the global routing system.

This feature enables communication between LISP sites that are connected to different RLOC spaces and have no connectivity to each other.

7.1.2 Prerequisites for LISP Support for Disjointed RLOC Domains

- You understand how LISP works, including infrastructure, workflow, roles and functions.

7.1.3 Information About LISP Support for Disjointed RLOC Domains

The fundamental principle of any network is that routing and reachability should exist between all devices that make up the total network system. There are many network systems, public and private, for which internetwork connectivity is not directly available.

- A Multiprotocol Label Switching (MPLS) IPv4 VPN from service provider A and an MPLS IPv4 VPN from service provider B, with different scopes, 10.1.0.0/16 and 10.2.0.0/16.
- An MPLS IPv4 VPN from service provider A and IPv4 internet.

When some sites within a network connect to one routing domain and other sites connect to another routing domain, a gateway function must be provided to facilitate connectivity between these disjointed routing domains. In traditional routing architectures, providing connectivity between disjointed routing domains can be quite complex.

The inherent property of LISP, which separates IP addresses into two address spaces, gives it the ability to connect disjointed RLOC domains through simplified configuration mechanisms. The key components are new control plane configuration options on the LISP Map-Server, and the Re-encapsulating Tunnel Router (RTR) function, which provides data plane connectivity between disjointed locator spaces. The components and the workflow are explained.

LISP Map-Server

LISP RTR

When a LISP site registers with the Map-Server, it provides RLOC information. Ensure that all relevant RLOCs are registered with the Map-Server. Map-Server configurations are required to enable connectivity across RLOC spaces.

An RTR provides data plane communications support for LISP to LISP traffic between LISP sites that do not share common locator space. Functionally, an RTR takes in LISP encapsulated packets from an Ingress Tunnel Router (ITR) in one locator scope, decapsulates them, checks the map-cache, and then re-encapsulates them to an Egress Tunnel Router (ETR) in another locator scope. The following are important considerations for an RTR:

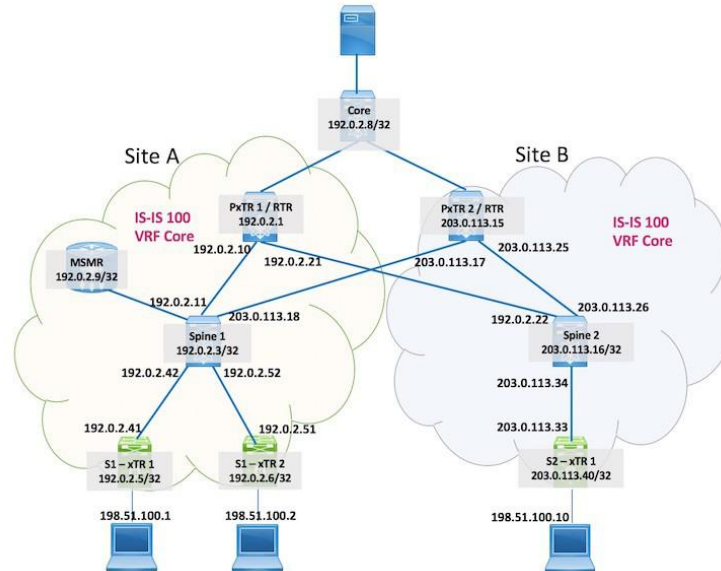
- RTR should have RLOCs in all locator scopes that are being joined.

- An RTR sends Map-Request messages to populate its map-cache. As a Map-Request message contains an ITR RLOC field that is populated with one or more entries corresponding to the locators of the device sending the Map-Request message, locator set configuration is required on the RTR to define its locators. This enables the Map-Server to correctly receive Map-Request messages from the RTR to assess locator scope connectivity.

- Since an RTR performs functions similar to a Proxy Ingress Tunnel Router (PITR) and Proxy Egress Tunnel Router (PETR), the PITR and PETR features must be enabled on the RTR.

Workflow of LISP Support for Disjointed RLOC Domains

Figure 24 : Sample topology—LISP Support for Disjointed RLOC Domains



For connecting disjointed RLOC domains in topology:

- Specified prefixes form the EID space in site A and site B.
- Ingress and Egress tunnel routers (referred as xTRs) represent the LISP site routers. xTR 1 and xTR 2 in site A have RLOC connectivity to one locator space, and the xTR in site B has RLOC connectivity to a different locator space.
- The RTR (PxTR 1, PxTR 2) is the LISP data plane device that enables communication between end hosts in the two sites, across locator spaces.
- Two virtual routing and forwarding (VRF) instances are created on the RTRs, one for the underlay (VRF core), and one for the overlay (VRF *vrf5000*).

An end host connected to xTR 1 in site A sends traffic to an end host attached to the xTR in site B. Since the source and destination RLOCs are from different RLOC spaces, PxTR 1 performs the role of RTR to transport traffic across the RLOC spaces. The detailed workflow:

1. xTR 1 (acting as an ITR) receives traffic from an attached end host, and sends a Map-Request for the destination EID (198.51.100.10), to the Map-Server (denoted by the IP address 192.0.2.9/32).
2. The Map-Server responds with a proxy reply containing the configured RTR locators (with IP addresses 192.0.2.1 and 203.0.113.15). The Map-Server does because the ITR-RLOC in the Map-Request from xTR 1 contains the RLOC from site A.
3. xTR 1 populates its map-cache with locator information (that is, PxTR 1 and PxTR 2 RLOCs) for the RTRs.
4. xTR 1 encapsulates LISP traffic and forwards it to the RTR in the data plane.
5. The RTR decapsulates the ingress LISP traffic and sends a Map-Request to the Map-Server for the destination EID, for the first packet.

6. The ITR-RLOC of the Map-Request comprises the locators configured under the locator set. The locators are 192.0.2.10 and 192.0.2.21. A Map-Request is sent because the static map-cache is configured with the map-request command.

7. The Map-Server forwards the Map-Request to the ETR. The Map-Server does because the ITR-RLOC in the Map-Request from the RTR contains RLOCs from site A and site B.

8. The ETR replies to the RTR with the ETR locator information.

9. The RTR populates its map-cache with the ETR locator information.

10. The RTR re-encapsulates LISP traffic forwards the ETR.

11. The ETR receives and sends traffic to the destination end host.

7.1.4 How to Configure LISP Support for Disjointed RLOC Domains

RTR configuration on PxTR 1 and the Map-Server:

PxTR 1 or RTR Configuration

Step 1 Configure LISP

```
(config)# feature lisp
```

Step 2 Create two VRF instances on the RTR, one for the underlay (VRF core), and one for the overlay (VRF vrf5000).

Configure LISP parameters for the core VRF

```
(config)# vrf context core
ip lisp itr-etr
ip lisp itr map-resolver 192.0.2.9/32
ip lisp etr map-server 192.0.2.9/32 key 3
a97b0defe7b8ff70 ip lisp multicast
lisp encapsulation vxlan
```

- After configuring the LISP ITR and ETR functions on PxTR 1, the LISP Map-Resolver (used by the ITR to send Map-Requests) and Map-Server (used by the ETR to register EIDs) locator addresses are configured.

- Also, LISP multicast transport and LISP Virtual Extensible LAN (VXLAN) encapsulation functions are enabled.

Configure LISP parameters for the vrf5000 VRF

```
(config)# vrf context vrf5000
ip lisp proxy-itr
192.0.2.1 ip lisp
proxy-etr
lisp instance-id 5000
```

The following configuration chunk is specific to connecting disjointed RLOC spaces.

```
lisp locator-set set5000
192.0.2.10 priority 1 weight 10
192.0.2.21 priority 2
weight 20 exit
lisp map-request itr-rlocs
set5000 ip lisp locator-
vrf core
ip lisp map-cache 198.51.100.1/24
map-request ip lisp map-cache
198.51.100.2/24 map-request ip lisp
multicast
lisp encapsulation vxlan
```

- The **lisp locator-set** command specifies a locator set for RTR RLOCs. 192.0.2.10 and 192.0.2.21 are the RLOCs connecting the RTR to each IPv4 locator space.
- The **lisp map-request itr-rlocs** command defines RTR RLOCs used in the Map-Request messages generated by the RTR. You can enable multiple locator sets, but only one of them can be active at a point in time, and that is determined by including the name in the **lisp map-request itr-rlocs** option.
- Since Map-Resolver and Map-Server addresses are enabled in VRF core, VRF core is referenced within VRF vrf5000, in the **locator-vrf core** command.

Step 3 Configure an IP address for routing in the underlay

```
(config)# interface loopback0
      vrf member core
      ip      address
      192.0.2.1/32  isis
      circuit-type level-
      1-2 ip router isis
      100
      ip pim sparse-mode
```

The configured loopback interface IP address is used for IS-IS communication within the LISP site, and is added to VRF core.

Step 4 The configurations are relevant for RTR or PxTR 1. Similarly, configure the RTR or PxTR 2 device too.

PxTR 1 or RTR Configuration—RTR Locator-Set Inheritance

An RTR locator set can be defined in the underlay VRF and can then be referenced in an overlay VRF.

Step 1 Configure LISP

```
(config)# feature lisp
```

Step 2 Create two VRF instances on the RTR, one for the underlay (VRF core), and one for the overlay (VRF vrf5000).

Configure LISP parameters for the *core* VRF

```
(config)# vrf context core
      ip lisp itr-etr
      ip      lisp      itr      map-resolver
      192.0.2.9/32
      ip lisp etr map-server 192.0.2.9/32 key 3
      a97b0defe7b8ff70 lisp locator-set setCore
      192.0.2.10 priority 1 weight
      10
      192.0.2.21 priority
      2 weight 20 exit
      ip lisp multicast
      lisp encapsulation vxlan
```

Configure LISP parameters for the *vrf5000* VRF

```
(config)# vrf context vrf5000
      ip lisp proxy-
      itr 192.0.2.1 ip
      lisp proxy-etr
      lisp instance-id 5000
      lisp map-request itr-
```

```

rlocs setCore ip lisp
locator-vrf core
ip lisp map-cache 198.51.100.1/24
map-request ip lisp map-cache
198.51.100.2/24 map-request ip
lisp multicast
lisp encapsulation vxlan

```

Step 3 Configure an IP address for routing in the underlay

```

(config)# interface loopback0
vrf member core
ip address
192.0.2.1/32
isis circuit-
type level-1-2
ip router isis
100
ip pim sparse-mode

```

The configured loopback interface IP address is used for IS-IS communication within the LISP site, and is added to VRF core.

Step 4 The configurations are relevant for RTR or PxTR 1. Similarly, configure the RTR or PxTR 2 device too.

Map-Server Configuration

Map-Server configuration on a device with IOS XE software (not Inspur CN12700 Series device):

```

(config)# router lisp
locator-table
vrf core
locator-set
SITEAB
192.0.2.1 priority 1
weight 50 203.0.113.15
priority 1 weight 50 exit
locator-scope site-B
rtr-locator-set SITEAB
rloc-prefix
203.0.113.40/32
rloc-prefix
192.0.2.21/32 rloc-
prefix
203.0.113.25/32
exit
locator-scope site-A
rtr-locator-set
SITEAB rloc-
prefix
192.0.2.5/32
rloc-prefix
192.0.2.6/32
rloc-prefix
203.0.113.17/32
rloc-prefix
192.0.2.10/32

```

7.1.5 Verifying LISP Support for Disjointed RLOC Domains

Testing Reachability from xTR 1 in Site A to the xTR in Site B

In the following example, locator information for both sites (192.0.2.1 in site A and 203.0.113.15 in site B) are displayed. xTR 1 in site A is connected to the xTR in site B.

```
siteA-xTR1# lig 198.51.100.10 instance-id 5000

Mapping information for EID 198.51.100.10 from 192.0.2.9/32 with RTT 2
msecs 198.51.100.10/32, uptime: 00:07:06, expires: 00:14:59, via map-
reply, complete

Locator          Uptime      State      Pri/Wgt
192.0.2.1       00:07:06   up         1/50
203.0.113.15   00:07:06   up         1/50
```

Testing Reachability from PxTR 1 to the xTR in Site B

In the following example, Map-Request, Map-Reply, and map-cache information is displayed. Also, locator information for the xTR in site B is displayed. This signifies that PxTR 1 is connected to the xTR in site B.

```
PxTR1# lig 198.51.100.10 vrf vrf5000

Send map-request to 192.0.2.9 for [5000]
100.3.3.10 ... Received map-reply from
203.0.113.40 with rtt 0.003248 secs Map-cache
entry for [5000] EID 198.51.100.10 is:
198.51.100.10/32, uptime: 05:05:47, expires: 23:59:58, via map-reply, auth

Locator          Uptime      State      Priority/Data  Control  MTU  Weight
in/out          in/out
10/10          0/0*   2/0  1500          203.0.113.40  05:05:47      up
```

EID Space Details in the Map-Server/Map-Resolver (MSMR)

In the following example, you can see that the client with the specified EID, attached to the xTR in site B, is registered with the MSMR. The specified EID, Instance ID and corresponding locator is displayed.

```
MSMR# show lisp site 198.51.100.10 instance-
id 5000

LISP Site Registration Information
Site name: site-AAllowed configured locators:
anyRequested      EID-prefix:      EID-prefix:
198.51.100.10/32 instance-id 5000
    First registered: 00:19:46
    Last registered:  00:19:46 Routing table tag: 0
    Origin:           Dynamic, more specific of 203.0.0.0/16
    Merge active:     No
    Proxy reply:      No
    TTL:              1d00h
    State:            complete
Registration
errors:
Authentication
failures: 0
Allowed locators mismatch: 0

ETR 203.0.113.40, last registered 00:19:46, no proxy-reply, map-notify
TTL 1d00h, no merge, hash-function sha1, nonce 0x4CC82237-
0x6DCB0FC5

state complete, no security-
capability
```



```

xTR-ID 0x90FA8033-0x867FE73F-0x5F32076D-0xE92E8945
site-ID unspecified
sourced by reliable transport

Locator      Local      State
             Pri/Wgt    Scope
203.0.113.40 yes      up
             10/10    site-B

```

In the following example, corresponding LISP site information for the MSMR is displayed. The information includes, EID, IID, and locator information.

```

MSMR# show lisp site detail

EID-prefix:      198.51.100.10/32
instance-id 5000 First registered:
                  08:12:10
Last registered: 08:12:10 Routing table tag:      0
Origin:          Dynamic, more specific of 203.0.0.0/16

Merge active:    No
Proxy reply:     No
TTL:            1d00h
State:          complete

Registration errors:
Authentication
failures:              0 Allowed locators mismatch: 0

ETR 203.0.113.40, last registered 08:12:10, no proxy-reply, map-notify
TTL 1d00h, no merge, hash-function sha1, nonce 0x4CC82237-
0x6DCB0FC5

state complete, no security-capability
xTR-ID 0x90FA8033-0x867FE73F-0x5F32076D-0xE92E8945
site-ID unspecified
sourced by reliable transport

Locator      Local  State Pri/Wgt
Scope 203.0.113.40 yes    up      10/10    site-B

```

Verify LISP map-cache Details on PxTR 1

In the following example, map-cache details corresponding to PxTR 1 for the specified EID are displayed. The information includes locator information.

```

PxTR1# show ip lisp map-cache 198.51.100.1 vrf vrf5001

LISP IP Mapping Cache for VRF "vrf5001" (iid 5001), 16 entries
* = Locator data counters are cumulative across all EID-prefixes

198.51.100.1/32, uptime: 1d03h, expires: 20:01:07, via map-
reply, auth Last activity: 03:58:42
State: complete, last modified: 1d03h, map-source:
192.0.2.5 Pending hw update: FALSE
Locator      Uptime    State      Priority/  Data      Control    MTU
             in/out    in/out    Weight    in/out    in/out
192.0.2.5    1d03h    up         10/10     0/0*     2/0       1500

Last up/down state change:      1d03h, state change
count: 0 Last data packet in/out: never/1d03h
Last control packet in/out:

```

52/never Last priority/weight 03:58:
change:
never/never

7.1.6 Feature History for LISP Support for Disjointed RLOC Domains

This table lists the release history for this feature.

Table 6 : Feature History for LISP Support for Disjointed RLOC Domains

Feature Name	Release	Feature Information	
Connecting LISP Disjointed RLOC Domains	8.4(1)	This feature was introduced.	

CHAPTER 8 PBR support for the VXLAN BGP EVPN fabric

Policy-based routing (PBR) support is provided for the Virtual eXtensible Local Area Network (VXLAN) Border Gateway Protocol (BGP) Ethernet VPN (EVPN) fabric. PBR allows you to configure a defined policy for IPv4 and IPv6 traffic flows, lessening reliance on routes derived from routing protocols. All packets received on a PBR enabled interface are passed through enhanced packet filters or route maps. The route maps dictate the policy, determining where to forward packets. PBR configurations have to be enabled on relevant Top of Rack (ToR) or leaf switches, and spine switches in the VXLAN BGP EVPN fabric.

This chapter contains the following sections:

- Prerequisites for PBR Support for the VXLAN BGP EVPN Fabric.
- Guidelines and Limitations for PBR Support for the VXLAN BGP EVPN Fabric .
- Information About PBR Support for the VXLAN BGP EVPN Fabric.
- How to Configure PBR Support for the VXLAN BGP EVPN Fabric.
- Verifying PBR Support for VXLAN BGP EVPN Fabric.
- Feature History for PBR Support for the VXLAN BGP EVPN Fabric.

8.1 Prerequisites for PBR Support for the VXLAN BGP EVPN Fabric

- A Inspur CN12700 Series switch with an F3 line card.
- Understand how VXLAN BGP EVPN works.
- Enable VXLAN BGP EVPN configurations on the leaf and spine switches.

8.2 Guidelines and Limitations for PBR Support for the VXLAN BGP EVPN Fabric

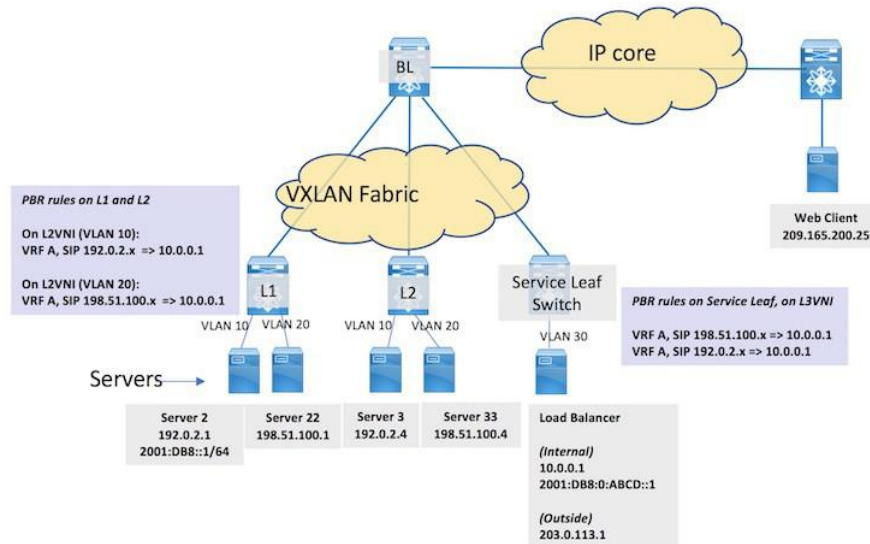
- This feature is not supported on port channels, channel members, Virtual Station Interface (VSI) trunk ports, and tunnel interfaces.
- Equal-Cost Multipath (ECMP) for IPv4/IPv6 adjacencies and BGP EVPN tunnel adjacencies are not supported at the same time.
- Virtual routing and forwarding (VRF) leak scenarios require that VRF context definitions should be consistent across all participating VTEPs.

8.3 Information About PBR Support for the VXLAN BGP EVPN Fabric

A sample workflow and PBR rules on ToR switches:

8.3.1 Workflow of PBR Support for the VXLAN BGP EVPN Fabric

Figure 25: Sample topology—PBR support for the VXLAN BGP EVPN fabric



Servers, virtual machines (VMs) and containers specific to a service are attached to ToR or leaf switches spread across the fabric (Server 2 and Server 22 attached to L1, and Server 3 and Server 33 attached to L2). The load balancer is attached to a service leaf switch at the right. A web client outside the fabric requests the service. When the request reaches the load balancer through the border leaf switch BL and the service leaf switch, it forwards the service request to an appropriate server (Server 2). PBR configurations should be enabled on L1, L2, and the service leaf switch. The workflow:

1. The web client (IP address 209.165.200.25) sends a request for a specific service available in the servers attached to L1 and L2. The destination IP address in the packets is 203.0.113.1, the load balancer's IP address. This IP address is advertised to clients outside the fabric. The request reaches BL.
2. BL knows from the BGP EVPN control plane that the load balancer is attached to the service leaf switch. BL VXLAN encapsulates the packets and forwards the request to the service leaf switch.
3. The service leaf switch receives it, VXLAN decapsulates the packets and sends the traffic to the load balancer.
4. The load balancer uses a unique IP address 10.0.0.1 (or 2001:DB8:0:ABCD::1 for IPv6) when load balancing traffic to one of the servers hosting the service. The load balancer decides to forward the request to Server 2. In the packet header, the load balancer updates the destination IP address to 192.0.2.1 (or 2001:DB8::1), while the source IP address remains 209.165.200.25 (or a designated IPv6 address), and forwards it to the service leaf switch. The service leaf switch sends the traffic over the VXLAN BGP EVPN fabric towards L1. L1 decapsulates the VXLAN header and forwards the original packet to Server 2.
5. Server 2 responds to the service request. The source IP address is 192.0.2.1 or 2001:DB8::1/64 (Server 2's IP address), and the destination IP address remains 209.165.200.25. The response is sent to L1.
6. On L1, normal packet forwarding determines that the destination address is behind the border leaf switch. However, PBR policy is applied on the interface on which the server is attached such that the packet is forwarded to the service leaf switch instead of the border leaf switch.
7. On the service leaf switch, after VXLAN decapsulation, normal packet forwarding determines that the destination address 209.165.200.25 is behind the border leaf switch. However, PBR policy is applied on the Layer-3 VNI interface such that the packet is forwarded to the service leaf switch instead of the border leaf switch.
8. The service leaf switch forwards it to BL, which forwards the service response to the web client.

8.3.2 PBR Rules on L1 and L2 (Enabled on BDI10 and BDI20)

- For server traffic (with source IP address 192.0.2.x, belonging to VLAN 10, VRF A, and Layer 2 virtual network identifier [VNI] 1000) received on interface BDI10, send traffic to 10.0.0.1 (or 2001:DB8:0:ABCD::1).
- For server traffic (with source IP address 198.51.100.x, belonging to VLAN 20, VRF A and Layer 2 VNI 2000), send traffic to 10.0.0.1 (or 2001:DB8:0:ABCD::1).

8.3.3 PBR Rules on the Service Leaf Switch

For server traffic (with source IP address 192.0.2.x or 198.51.100.x, belonging to VRF A, and Layer 3 VNI 50000) received on the PBR interface, send traffic to 10.0.0.1 (or 2001:DB8:0:ABCD::1).

8.4 How to Configure PBR Support for the VXLAN BGP EVPN Fabric.

8.4.1 Enable PBR Configurations on L1, L2 and the Service Leaf Switch

Step 1 On L1, configure PBR and create an access list:

```
(config)# feature pbr
ip access-list 1
  permit ip 192.0.2.1 0.0.0.255
  host 10.0.0.1 exit
ipv6 access list vlan102-112
10 permit ipv6 2001:DB8::1/64 2001:DB8:0:ABCD::1/64
```

- As per PBR rules, traffic from 192.0.2.1 or 2001:DB8::1/64 (Server 2's IP address) is sent to 10.0.0.1 or 2001:DB8:0:ABCD::1/64 (the load balancer's IP address), respectively.

Step 2 On L1, create a route map and set rules:

```
(config)# route-map equal-
access permit 10 match ip
address 1
      set ip vrf A next-hop 10.0.0.1

(config)# route-map equal-access-
v6 permit 10 match
ipv6 address vlan102-
112
set ip vrf A next-hop
2001:DB8:0:ABCD::1
```

- An IPv4 route map policy equal-access, and an IPv6 route map policy equal-access-v6 are created.
- The **set ip vrf** command (introduced in this feature) resolves the next hop IP address for VRF A, for the IPv4 and IPv6 route map policies.

Step 3 Enable the route map policy on the BDI:

```
(config)# interface bdi10
vrf member A
ip policy route-map
equal-access ipv6 policy
route-map equal-access-v6
```

Step 4 The configurations are relevant for BDI10. Similarly, enable PBR configurations for BDI 20 on L1.

Step 5 Enable PBR configurations on BDI10 and BDI20 on L2 and service leaf switch.

8.5 Verifying PBR Support for VXLAN BGP EVPN Fabric

Verify PBR configurations on L1:

8.5.1 Verifying PBR Route Map Policy Configuration on the BDI

PBR route map policy equal-access is implemented on BDI10:

```
L1# show running-configuration interface Bdi10
...
interface Bdi10 no shutdown vrf member A
no ip redirects
ip address 192.0.2.200/24
fabric forwarding mode
anycast-gateway ip policy
route-map equal-access
ipv6 policy route-map equal-access-v6
```

8.5.2 Verifying PBR Route Map Policy Details

Route map policy configuration details:

```
L1# show running-configuration rpm

feature pbr
route-map equal-access
  permit 10 match ip
  address 1
  set ip next-hop 10.0.0.1

route-map equal-access-v6
  permit 10 match ipv6
  address vlan102-112
  set ipv6 next-hop 2001:DB8:0:ABCD::1

interface Bdi100
  ip policy route-map equal-
  access ipv6 policy route-map
  equal-access-v6
```

8.6 Feature History for PBR Support for the VXLAN BGPEVPNFabric

This table lists the release history for this feature.

Table 7 : Feature History for PBR Support for the VXLAN BGP EVPN fabric

Feature Name	Release	Feature Information	
PBR support for the VXLAN BGP EVPN fabric	8.4(1)	This feature was introduced.	

CHAPTER 9 VXLAN BGP EVPN and OTV Interoperation

A datacenter fabric can be Virtual eXtensible Local Area Network (VXLAN) Border Gateway Protocol (BGP) Ethernet VPN (EVPN) based, classical ethernet (CE) based, or combined. You can connect different datacenters over an IP WAN with Layer 2 (such as Overlay Transport Virtualization [OTV]) and Layer 3 technologies, and also connect end hosts between Layer 2 CE and VXLAN pods within a datacenter. With this feature, you can:

- Configure VXLAN and OTV (with Bridge Domain Interface [BDI]) as a one-box solution. The VXLAN and OTV overlays or tunnels are stitched together on a VXLAN BGP EVPN fabric border leaf switch, ensuring that the Layer 2 traffic between VXLAN and OTV is within the same bridge domain.

- Configure OTV with BDI in a datacenter as a one-box solution, instead of a two box solution used in a legacy datacenter.

OTV as the overlay enables Layer 2 connectivity between separate VXLAN BGP EVPN or CE Layer 2 domains while maintaining resiliency and load-balancing benefits of the IP based interconnection.

- Prerequisites for VXLAN BGP EVPN and OTV Interoperation.
- Guidelines and Limitations for VXLAN BGP EVPN and OTV Interoperation .
- Information About VXLAN BGP EVPN and OTV Interoperation.
- How to Configure VXLAN BGP EVPN and OTV Interoperation, and OTV with BDI.
- Verifying VXLAN BGP EVPN and OTV Interoperation, and OTV with BDI.
- Feature History for VXLAN BGP EVPN and OTV Interoperation .

9.1 Prerequisites for VXLAN BGP EVPN and OTV Interoperation

- A CN12700 Series switch with an F3 line card.
- Conceptual and configuration knowledge about VXLAN BGP EVPN and OTV datacenter fabrics.
- For a functioning VXLAN BGP EVPN datacenter, configurations should be enabled on the leaf and spine switches..

9.2 Guidelines and Limitations for VXLAN BGP EVPN and OTV Interoperation

- A unique, primary gateway IP address should be configured on each BDI for sending ARP requests over OTV. A secondary anycast IP address is configured on each BDI for sending ARP requests over EVPN.

- For Layer 3 multicast traffic, an external, centralized Layer 3 multicast gateway should be enabled. Layer 3 multicast routing support is not available for the VXLAN BGP EVPN and OTV Interoperation feature in release 8.4(1). However, Layer 3 multicast traffic within a VXLAN BGP EVPN fabric is supported as usual.

- For seamless mobility across legacy and VXLAN BGP EVPN fabrics, the anycast gateway MAC address used in the VXLAN fabric should be configured as the HSRP static MAC address in the legacy fabric.

- Only OTV unicast control network (OTV Adjacency Server function) is supported.

- On the same physical Join interface, regular OTV overlays cannot interoperate with OTV overlays that are stitched to VXLAN overlays. However, an OTV overlay and an OTV+VXLAN overlay can be enables on separate physical Join interfaces.

- In an OTV with BDI single box solution, the ARP proxy function option is not supported in the Inspur INOS 8.4(1) release.

- Support for this feature is limited to 3 OTV datacenter sites, for the 8.4(1) release.

- For OTV overlays, only Generic Routing Encapsulation (GRE) encapsulation is supported for the 8.4(1)

release.

9.3 Information About VXLAN BGP EVPN and OTV Interoperation

Two VXLAN BGP EVPN datacenters can be connected with each other and to a legacy datacenter (Sample topology 1). A VXLAN BGP EVPN datacenter can be connected to a legacy datacenter and to a datacenter with a OTV+BDI one box solution implemented in the form of a vPC pair of border switches (Sample topology 2). In either topology, the Layer 3 aggregation switch in the legacy datacenter performs the role of an external, centralized multicast gateway for Layer 3 multicast traffic across the datacenters. The two sample topologies and workflow for the VXLAN BGP EVPN and OTV interoperation in them:

9.3.1 Sample Topologies and Workflow of the VXLAN BGP EVPN and OTV Interoperation

Figure 26 : Sample topology 1 - VXLAN BGP EVPN and OTV interoperation

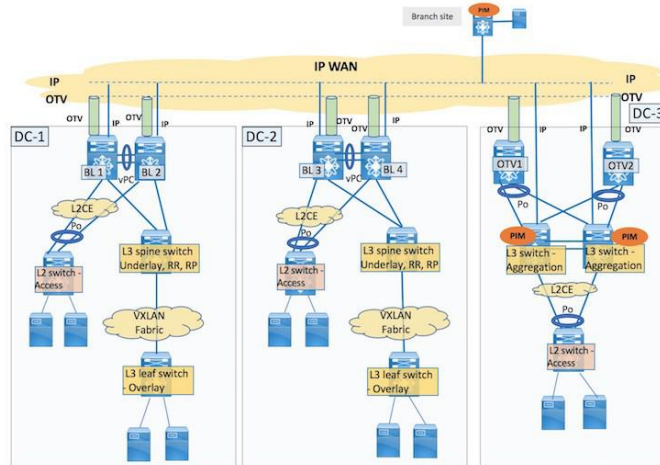
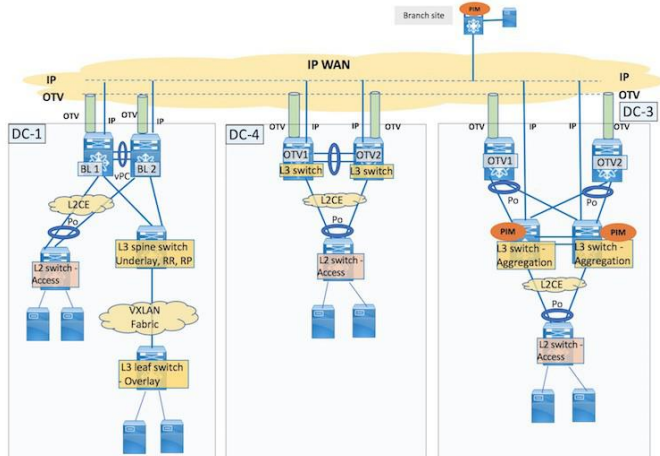


Figure 27 : Sample topology 2 - VXLAN BGP EVPN and OTV interoperation



The focus of this feature is the VXLAN+OTV interoperation and OTV+BDI function on a CN12700 Series border leaf switch with an F3 line card, and the traffic flow between servers in these and a legacy datacenter. Topology information:

- Topology 1 displays 3 datacenters, DC-1, DC-2 and DC-3. DC-1 and DC-2 have VXLAN BGP EVPN and CE pods, with VXLAN and OTV (with Bridge Domain Interface [BDI]) configurations on the Virtual Port Channel (vPC) pair border leaf switches (BL 1 and BL 2, BL 3 and BL 4). DC-3 is a legacy OTV site with OTV switches in separate VDCs (OTV1 and OTV2) at the border.

- Topology 2 displays 3 datacenters, DC-1, DC-4 and DC-3. DC-1 has VXLAN BGP EVPN and CE pods, with VXLAN and OTV (with BDI) enabled on the vPC border leaf switch pair. DC-4 is a OTV+BDI one box solution with configurations implemented on the vPC border switch pair. DC-3 is a legacy OTV site.

9.3.2 Layer 2 Switching

Layer 2 traffic is transported between the datacenters through the border leaf switches (in DC-1 and DC-2) and OTV devices (in DC-3 and DC-4) at the site border, over the IP WAN.

In DC-1 and DC-2, there are 2 scenarios where Layer 2 traffic is transported between VXLAN and OTV overlays or tunnels. On the border leaf switch, traffic from the VXLAN fabric is either sent to a server within the datacenter, or towards another datacenter through OTV. The other way around, the border leaf switch receive traffic from OTV towards the VXLAN fabric or CE pod. When you configure this feature, Layer 2 traffic seamlessly passes between the VXLAN and OTV tunnels on the same device. Packet flow details in DC-1 and DC-2:

- Packet flow within the Layer 2 CE pod and packet flow within the VXLAN BGP EVPN pod (see "Configuring VXLAN BGP EVPN" chapter) remains the same. When a Layer 2 CE pod server sends traffic to a server in the VXLAN fabric within the site or to another site, the packets reach the border leaf switch. The bridge domain, Layer 2/Layer 3 VNI mappings, and MAC routes of the VXLAN fabric are available in the border leaf switch. If the destination server is within the fabric, the border leaf switch VXLAN encapsulates the packet and sends it to the corresponding ToR or leaf switch. The leaf switch VXLAN decapsulates the traffic and sends the original packet to the intended server. If the destination server is in another site, the border leaf switch OTV encapsulates the traffic towards the remote site.

- When a VXLAN+OTV border leaf switch receives traffic from another site over OTV, it removes the OTV encapsulation, does a lookup to find out where the destination server resides, and VXLAN encapsulates the traffic towards the corresponding ToR/leaf switch. The leaf switch VXLAN decapsulates the traffic and sends the original packet to the intended server. If the destination server is in the Layer 2 CE pod, the border leaf switch OTV decapsulates the traffic and sends the traffic to the destination server without any encapsulation.

- When a server in the VXLAN BGP EVPN fabric sends traffic to a server in the Layer 2 CE pod, the border leaf switch receives the packets. It VXLAN decapsulates the traffic and sends it to the destination server in the CE pod.

Packet flow details in DC-4:

- When a Layer 2 CE pod server in the datacenter with the OTV+BDI one box solution sends traffic, the destination server is either within the datacenter or outside of it. Traffic flow within the datacenter remains the same. If the destination server is in another site, then the packets reach the Layer 3 OTV (with BDI) switch. The switch OTV encapsulates the traffic towards the legacy or VXLAN+OTV datacenter. The border switch in the destination datacenter receives the traffic, OTV decapsulates it, and forwards it as explained in the earlier sections.

- When the Layer 3 OTV (with BDI) switch receives traffic from another datacenter over OTV, it OTV decapsulates the traffic and sends it towards the corresponding Layer 2 access switch. The access switch forwards the packets to the destination server.

9.3.3 Control Plane

- BGP EVPN is used for advertising MAC and MAC-IP routes across the VXLAN BGP EVPN fabric in DC-1 and DC-2.

- Intermediate-System to Intermediate-System (IS-IS) is used for advertising MAC routes across OTV configured device.

- MAC Route updates in the control plane are reflected across the OTV and VXLAN tunnels.

9.3.4 Layer 3 Unicast Routing

DC-1 and DC-2 (VXLAN BGP EVPN fabric)

Layer 3 routing within the VXLAN BGP EVPN fabric takes place through an underlay Interior Gateway Protocol (IGP) such as Intermediate System-to-Intermediate System (IS-IS) or OSPF.

Layer 3 traffic between the border leaf switches and the IP WAN should be through over Multiprotocol Label Switching (MPLS) L3VPN or virtual routing and forwarding (VRF) Lite. The IGP and external connectivity documentation is available in the *Inspur Programmable Fabric with VXLAN BGP EVPN Configuration Guide*.

A distributed anycast gateway (or BDI) IP address is used for Layer 3 traffic between Layer 2 virtual networks in the VXLAN fabric. This should be configured as the secondary BDI IP address. A unique, primary gateway IP address should be configured on each BDI for sending ARP requests over OTV. Packet flow details in DC-1 and DC-2:

- When a server in the Layer 2 CE pod sends traffic to a server in another VLAN/subnet, either located in the VXLAN BGP EVPN fabric within the site or to a server in a remote site, the traffic first reaches the border leaf switch. The bridge domain, Layer 2/Layer 3 VNI mappings, MAC routes and the appropriate IGP configuration will be available in the border leaf switch. If the destination server is within the fabric, the border leaf switch VXLAN encapsulates the packet and routes it to the corresponding ToR or leaf switch through the fabric underlay routing protocol such as IS-IS or OSPF. The leaf switch VXLAN decapsulates the traffic and sends the original packet to the intended server. If the destination server is in another site, the border leaf switch sends the traffic towards the remote site through a Layer 3 DCI technology like MPLS L3VPN or OTV, depending on how the host route is learnt (through MPLS VPN or OTV).
- When the Layer 3 DCI enabled border switch receives Layer 3 traffic from another site (through MPLS VPN or OTV), it does a lookup to find out where the destination server resides and routes the packets (across VLANs, through the corresponding destination BDI), to the corresponding Layer 2 switch. The switch forwards the packets to the intended destination server.

DC-3, the legacy datacenter

- Layer 3 routing within DC-3 is through an IGP implemented on the aggregation switches. A Layer 3 Hot Standby Router Protocol (HSRP) gateway with First Hop Redundancy Protocol (FHRP) filtering should be configured.
- Configure the anycast gateway MAC address used in the VXLAN fabric as the HSRP MAC address in the legacy fabric for efficient MAC moves.
- Layer 3 traffic between the legacy fabric and the IP WAN should be over OTV, MPLS L3VPN, or VRF Lite.

DC-4, the datacenter with the OTV+BDI switch as the one box solution

- Layer 3 routing within this datacenter is through an IGP implemented on the Layer 3 vPC switch pair at the border. When a server in DC-4 sends traffic to a server in another VLAN/subnet, either located within the site or in a remote site, the traffic first reaches the designated Layer-3 switch. If the destination is within the fabric, it routes the packets (across VLANs, through the corresponding BDI) towards the corresponding Layer 2 switch which forwards it to the destination. If the destination is in another site, the Layer-3 border switch routes the traffic towards the remote site, through a Layer 3 DCI technology like MPLS L3VPN or OTV, depending on how the host route was learnt (through MPLS VPN or OTV).
- When the Layer 3 DCI enabled border switch receives Layer 3 traffic from another site, it does a lookup to find out where the destination server resides and routes the traffic through the IGP in the fabric, to the corresponding Layer 2 switch. The switch forwards the packets to the intended destination server.

ARP requests

- All ARP requests are resolved through the VXLAN and/or OTV overlays.
- When ARP suppression is enabled on the VXLAN overlay NVE interface (for a specific Layer 2 virtual

network[s]), then an ARP request is suppressed if the entry is present in the ARP cache. Else, the request is flooded into the VXLAN fabric.

- A Layer 3 gateway generated ARP request sent over OTV uses the primary gateway IP and virtual device context (VDC) MAC addresses, while an ARP request sent over the VXLAN BGP EVPN fabric uses the secondary BDI or anycast gateway IP and corresponding MAC addresses.

9.3.5 Layer 2 Multicast Forwarding and Layer 3 Multicast Routing

Multicast traffic across the datacenters:

Layer 2 Multicast Forwarding

- When a server in the VXLAN BGP EVPN fabric sends multicast traffic to the attached ToR/leaf switch, the leaf switch forwards the multicast traffic within the fabric, as explained in the Inspur Programmable Fabric with VXLAN BGP EVPN Configuration Guide. If there are receivers in the Layer 2 CE pod within DC-1, the leaf switch sends traffic towards the border switch, which VXLAN decapsulates the traffic and sends bridged traffic to relevant Layer 2 switches (or receivers in the Layer 2 CE pod). For receivers in DC-2, DC-3, or DC-4, the border leaf switch VXLAN decapsulates the traffic and sends traffic over OTV to DC-2, DC-3 (or DC-4).

- When the DC-2 border leaf switch receives the packets, it has to send traffic to receivers in the VXLAN fabric and in the Layer 2 CE pod. For receivers within the VXLAN fabric, the border leaf switch OTV decapsulates the traffic, VXLAN encapsulates it, and sends it towards relevant ToR/leaf switches. For receivers in the CE pod, the traffic is OTV decapsulated and sent towards relevant Layer 2 CE switches.

- When a DC-3 OTV switch receives the packets, the switch OTV decapsulates the traffic and sends a copy of the bridged traffic towards relevant Layer 2 CE switches/receivers.

- When the designated DC-4 OTV+BDI border vPC switch receives the packets, it OTV decapsulates the traffic and sends a copy of the bridged traffic towards relevant Layer 2 CE switches/receivers.

- Similarly, when a server in the Layer 2 CE pod (in DC-1 or DC-2) sends multicast traffic, traffic flow to receivers within the CE pod remains the same. If there are receivers in the VXLAN fabric in DC-1, and in DC-2 and DC-3, the attached access switch sends it towards the border leaf switch. The border leaf switch sends traffic to receivers in the VXLAN fabric (in DC-1), and towards DC-2 and DC-3, as explained in the previous points. If there are receivers in DC-4 (per topology 2), a copy of the packets is sent towards DC-4.

- When a server in DC-4 (OTV+BDI one box solution at the border) sends Layer 2 multicast traffic, traffic flow to receivers within the CE pod remains the same. If there are receivers in other datacenters, the switch at the datacenter border receives the packets, and sends it over OTV towards the destination datacenters (DC-1 and DC-3). If DC-4 has multicast receivers and receives multicast traffic from a sender in another datacenter, it OTV decapsulates the traffic and sends a copy of the bridged traffic towards relevant Layer 2 CE switches/receivers. The Layer 2 switch(es) forwards the packets to intended destination servers.

Layer 3 multicast routing

When a server in DC-1 or DC-2 (in topology 1), or DC-1/DC-4 (in topology 2) sends Layer 3 multicast traffic across the fabrics, it is routed through the external, centralized multicast gateway in the legacy datacenter, DC-3. Layer 3 multicast routing support is not available for the VXLAN BGP EVPN and OTV Interoperation feature in release 8.4(1). On the border leaf switch, ensure that you do not enable the ip pim sparse-mode command on OTV enabled bridge domains.

Attention

Layer 3 multicast traffic within a VXLAN BGP EVPN fabric (for VXLAN enabled bridge domains) is supported as usual. This is the use case wherein a sender within the VXLAN fabric sends Layer 3 multicast traffic to receivers located within the fabric. For more details, see "Multicast Routing in the VXLAN Underlay" section, "IP Fabric Underlay" chapter in the Inspur Programmable Fabric with VXLAN BGP EVPN Configuration Guide.

Layer 3 multicast packet flow in DC-1 and DC-2:

- If a server in the VXLAN fabric in DC-1 sends multicast traffic to receivers in different VLANs/subnets

outside the VXLAN fabric, then the traffic is sent to the Layer 3 switch in DC-3 where the centralized multicast gateway is enabled. First, the traffic reaches the border leaf switch in DC-1. Assuming that a receiver is located in the CE pod in DC-1 and another receiver in the VXLAN fabric in DC-2, the border leaf switch forwards the traffic towards DC-3. On receiving the request, the Layer 3 switch (or centralized multicast gateway) in DC-3 sends a copy of the routed multicast traffic to the border leaf switch in DC-1 and DC-2. On receiving the traffic, the border leaf switch in DC-2 VXLAN encapsulates the traffic and sends it towards the corresponding ToR/leaf switch. The leaf switch VXLAN decapsulates the packets and sends it to the destination server. The border leaf switch in DC-1 forwards the traffic to the corresponding Layer 2 CE switch, which forwards it to the attached destination server.

Layer 3 multicast packet flow in DC-4:

- If a server in this datacenter sends multicast traffic to receivers in different VLANs/subnets, the traffic first reaches the Layer 3 OTV+BDI switch at the border. The Layer 3 switch forwards it towards DC-3, (the external, centralized multicast gateway) which sends a copy of the routed multicast traffic towards other datacenter fabrics that have receivers. The rest of the flow is the same as explained in the previous points.

9.4 How to Configure VXLAN BGP EVPN and OTV Interoperation, and OTV with BDI.

9.4.1 Configure VXLAN BGP EVPN and OTV Interoperation on the Border Leaf Switches in DC-1 and DC-2

Only configurations relevant to the VXLAN BGP EVPN and OTV Interoperation feature are noted here. If VXLAN BGP EVPN fabric configurations are not enabled on the fabric's leaf and spine switches, enable them. See the "Configuring the VXLAN BGP EVPN" chapter.

VXLAN BGP EVPN and OTV configurations should be configured on a single box, the border leaf vPC switch pair. BL1 and BL2 configurations on DC-1:

Step 1 Configure tunnel stitching in the VXLAN overlay

The VXLAN BGP EVPN configurations are documented in the Configuring VXLAN BGP EVPN chapter. Only VXLAN configurations required for the VXLAN and OTV interoperation for the one-box solution are given.

```
BL1(config)# feature nve
vni 40000
system bridge-domain 2500-3500
bridge-domain 3500
member
vni
40000
exit
interface nve1
source-interface
loopback0 tunnel-
stitching enable
member vni 40000
no suppress-arp
mcast-group 239.1.1.65
```

- The **tunnel-stitching enable** command is the VXLAN command for connecting VXLAN and OTV tunnels.

Step 2 Configure the VXLAN overlay loopback interface

```
BL1(config)# interface loopback0
ip address 209.165.200.25/32
ip address 203.0.113.1/32
```

```
secondary
```

- Primary and secondary IP addresses are assigned to the source loopback interface on the switch.

Step 3 Configure tunnel switching in the OTV overlay

```
BL1(config)# feature otv
otv site-vni 40000 interface Overlay1
    otv join-interface
    Ethernet5/5      otv
    extend-vni      10000,
    20000
    otv vni mapping 10000, 20000, 30000 to vlan 1000,
    2000, 3000 otv use-adjacency-server 10.0.0.1
    unicast-only
no otv suppress-arp-nd otv tunnel-stitch
no shutdown exit
otv      site-identifier
0000.0000.000A      otv
encapsulation-format ip
gre
```

- The **otv site-vni** command enables the OTV site specific VNI. This VNI should not be extended over any overlay interface and should be operationally up before it can be configured as the OTV site VNI. At least one interface should be present where the VNI is up. On a VXLAN + OTV pod, the VNI can be configured under the NVE interface. If the **otv site-vlan** configuration is enabled, then you need to remove it before configuring the **otv site-vni** command.
- The **otv tunnel-stitch** command is the OTV command for connecting VXLAN and OTV tunnels.
- For OTV overlays, only Generic Routing Encapsulation (GRE) encapsulation is supported for the 8.4(1) release.

Step 4 Configure the VXLAN and OTV tunnel stitching configurations on BL2.

Step 5 Configure vPC function on BL1 and BL2 vPC Peer 1 (BL1) configuration

```
BL1(config)# interface Bdi3500
no shutdown
vrf member      cust1      no      ip
redirects
ip address 198.51.100.20/24
ip address 198.51.100.1/24      secondary
anycast-primary      ipv6      address
2001:DB8:1::1/64
no ipv6 redirects
fabric forwarding mode anycast-gateway
```

vPC Peer 2 (BL2) configuration

```
BL2(config)# interface Bdi3500
no shutdown
vrf
member
cust1 no
ip
redirect
s
ip address 198.51.100.30/24
ip address 198.51.100.1/24      secondary
anycast-primary      ipv6      address
2001:DB8:1::1/64
no ipv6 redirects
fabric forwarding mode anycast-gateway
```

- The unique, primary gateway IP address (198.51.100.20 for vPC peer 1, and 198.51.100.30 for vPC peer 2

switches) will be used for sending ARP requests over OTV. The common, secondary anycast gateway IP address (198.51.100.1) will be used for sending ARP requests on the VXLAN side.

Step 6 The configurations are for DC-1. Similarly, configure for DC-2 too.

9.4.2 Configure OTV with BDI Configuration on the Border Switches in DC-4

OTV1 and OTV2 configurations:

Step 1 Configure OTV on OTV1

```
OTV1(config)# feature otv
                feature      nv
                overlay      feature-
                set          fabric
                feature      fabric
                forwarding
                feature vni
                vni 40000
                system bridge-domain 2500-3500
                bridge-domain 3500
                    member
                    vni
                    40000
                exit
                encapsulation profile vni
                vsi_Inspur dot1q 60 vni
                40000
                exit
                interface
                Ethernet
                1/12      no
                shutdown
                service instance 1 vni
                encapsulation          profile
                vsi_Inspur      default      no
                shutdown
```

The **feature nv overlay** command is a required command for this feature.

```
OTV1(config)# no otv site-vlan 111
                otv      site-vni
                40000
                interface
                Overlay1
                otv      join-interface
                Ethernet2/3 otv extend-
                vni 10000, 20000, 30000
                otv vni mapping 10000, 20000, 30000 to vlan 1000, 2001, 3000
                otv use-adjacency-server 10.0.0.1 192.0.2.1
                unicast-only no otv suppress-arp-nd
                otv      adjacency-server
                unicast-only      no
                shutdown
                exit
                otv      site-identifier
                0000.0000.000A      otv
                encapsulation-format ip
                gre
```

- The **otv site-vni** command enables the OTV site specific VNI. This VNI should not be extended over any overlay

interface and should be operationally up before it can be configured as the OTV site VNI. At least one interface should be present where the VNI is up. On a OTV with BDI pod, the VNI can be brought up on a VSI interface, using an encapsulation profile, as shown in the example. If the **otv site-vlan** configuration is enabled, then you need to remove it before configuring the **otv site-vni** command.

- In an OTV with BDI single box solution, the **otv suppress-arp-nd** option is not supported in the Inspur INOS 8.4(1) release.
- For OTV overlays, only Generic Routing Encapsulation (GRE) encapsulation is supported for the 8.4(1) release.

Step 2 Similarly, enable OTV configurations on OTV2.

Step 3 Configure vPC function on OTV1 and OTV2 vPC Peer 1 (OTV1) configuration

```
OTV1(config)# interface Bdi3500
                no shutdown
                vrf member cust1 no ip
                redirects
                ip address 209.165.201.10/24
                ip address 209.165.201.20/24 secondary
                anycast-primary fabric forwarding mode
                anycast-gateway
```

vPC Peer 1 (OTV2) configuration

```
OTV1(config)# interface Bdi3500
                no shutdown
                vrf member cust1 no ip
                redirects
                ip address 209.165.201.12/24
                ip address 209.165.201.20/24 secondary
                anycast-primary fabric forwarding mode
                anycast-gateway
```

- The primary, unique IP address (209.165.201.10 for vPC peer 1 switch [OTV1], and 209.165.201.12 for vPC peer 2 switch [OTV2]) is created for ARP requests over OTV. The common, secondary anycast gateway IP address (209.165.201.20) is used as a Layer 3 gateway for traffic, including ARP requests, from the specified bridge domain Bdi3500 within the datacenter.

9.5 Verifying VXLAN BGP EVPN and OTV Interoperation, and OTV with BDI

Verify VXLAN and OTV configurations on BL1:

9.5.1 Display VXLAN Configuration on the Border Leaf Switch BL1

In the following example, you can see the VXLAN overlay information such as the loopback interface information, and the state of the overlay. For vPC switches, primary and secondary IP addresses are configured:

```
BL1# show nve interface

Interface:    nve1,      State:    Up,
encapsulation: VXLAN VPC Capability:
VPC-VIP-Only [not-notified] Local Router
MAC: 1005.caf4.cdd9
Host Learning Mode: Control-Plane
Source-Interface: loopback1 (primary: 209.165.200.25, secondary: 203.0.113.1)
```

In the following example, you can view bridge domain information for the specified bridge domain:

```
BL1# show bridge-domain 100
```

```
Bridge-domain 100 (2 ports
in all) Name: Bridge-
Domain100
Administrative State: UP
Operational State: UP
vni10000
Overlay1 nve1
```

In the following example, you can see VNI information for the VXLAN overlay, including Layer 2/Layer 3 VNI, multicast group mapping, and so on:

```
BL1# show nve vni
```

```
Codes: CP - Control Plane      DP - Data
       Plane UC - Unconfigured SA -
       Suppress ARP

Interface VNI      Multicast-group  State Mode Type [BD/VRF]      Flags
-----
-- ----- nve1    5000             224.1.1.1 Up    CP    L3
[cust1]
```

In the following example, you can see attached VXLAN overlay peer information such as the peer IP address, state, and so on:

```
BL1# show nve peers
```

```
Interface Peer-IP      State  LearnType  Uptime  Router-Mac
-----
nve1      10.1.1.2    Up     CP         4d19h   8c60.4f04.21c2
```

9.5.2 Display OTV Configuration on the Border Leaf Switch BL1

In the following example, you can see the extended VLANs and VNIs in the **Extended vlans** and **Extended vni** fields. Also, the OTV join interface and adjacency server information are displayed:

```
BL1# show otv
```

```
OTV Overlay Information
Site Identifier
0000.0000.000a
Encapsulation-
Format ip - gre

Overlay interface Overlay1

VPN name          : Overlay1
VPN state         : UP
Extended vlans    : 50 60 100-101 300-301
(Total:6)
Extended vni      : 10000, 20000, 30000,
40000(total:4) Join interface(s) :
Eth5/5 (10.3.3.3)
Site vlan         : 101 (up)
AED-Capable      : Yes
Capability        :
Unicast-Only     : Is
```



```
Adjacency Server : No
Adjacency Server(s) : 10.0.0.1 / [None]
```

In the following example, site VLAN, site VNI, and other site details are displayed. You should remove the site VLAN using the **no otv site-vlan** command.

```
BL1# show otv site detail

Full      - Both site and overlay
adjacency up Partial  - Either
site/overlay adjacency down
Down      - Both adjacencies are down (Neighbor is
down/unreachable) (!)      - Site-ID mismatch detected

Local Edge Device
Information:
Hostname BL1
System-ID 002a.6a65.09c1
Site-Identifier
0000.0000.000a
Site-VLAN      101
State is Up

Site-VLAN-Cfg 0
Site-VNI 10001

Site Information for Overlay1:

Bridge-domain 101 (1 ports
in all) Name:: Bridge-
Domain101
Administrative          State:          UP
                        Operation
                        al State: UP vni10001
nvel
```

9.5.2 Display OTV Overlay State

In the following example, the **Tunnel Stitch** field displays **enabled** state, indicating that the OTV part of the VXLAN OTV tunnel stitching is enabled:

```
BL1# show otv internal overlay detail

Overlay interface Overlay1 idx:
1635021663 Overlay interface idx:
0x22080001
VPN name          : Overlay1
Device ID         : 8c60.4f04.21c6
Protocol state    : UP           Interface state    : UP
Tunnel Stitch     : enabled
Tier Id           : 1
Pend Tier Id      : 0
Tier Id State     : Done(4)
Refcount          : 6
```

In the following example, the overlay ID and the VXLAN-to-OTV tunnel stitching status are displayed. In the **Enable** column, Y indicates that the OTV part of the tunnel is enabled:

```
BL1# show otv internal tunnel-stitch

Overlay  Enable Tier-id  Tier-pend-id  State  Detail
-----  -
Overlay1  Y  1      0      4      Done
```

In the following example, you can see that ARP proxy is enabled under the OTV overlay:

```
BL1# show otv internal arp-nd status

Overlay: Overlay1
  Suppress arp-nd:
    Enabled      VNI
  Suppress ARP:
50000  1000      : Enabled
50001  1001      : Enabled
50002  1002      : Enabled
50003  1003      : Enabled
50004  1004      : Enabled
50005  1005      : Enabled
50006  1006      : Enabled
50007  1007      : Enabled
```

In the following example, for the specified OTV overlay (Overlay1), VNI, BD, and VLAN mapping information is displayed. The **State** field indicates that the overlay is functional.

```
BL1# show otv vni Overlay1

Overlay  VNI      BD    VLAN  Flag  State
-----  -
Overlay1 10000  100   1000  0x1   (12)Overlay Logical up
Overlay1 20000  101   2000  0x1   (12)Overlay Logical up
Overlay1 30000  300   3000  0x1   (12)Overlay Logical up
Overlay1 40000  301   3500  0x1   (12)Overlay Logical up
```

9.5.4 Display OTV Adjacencies

In the following example, OTV adjacencies towards the legacy datacenter switches OTV1 and OTV2 in DC-3, BL2 in DC-1, and BL3 in DC-2 are displayed:

```
BL1# show otv
adjacency Overlay

Adjacency

Hostname Database System ID      Dest Addr      Up Time  State
-----
LEGACY-OTV1 002a.6a65.0045 209.165.200.25 3d01h    UP
LEGACY-OTV2 002a.6a65.0046 203.0.113.1    3d01h    UP
core-Bl2   8c60.4f04.21c7 203.0.113.200 3d01h    UP
BL3        8c60.4f04.43c1 203.0.113.250 3d01h    UP
```

9.5.5 Display Tier IDs

In the following example, you can see tier IDs allocated to the VXLAN and OTV overlays. Unique tier IDs are automatically allocated so that packets that are sent from a tunnel are not sent back to the same tunnel, such as multicast packets being sent back to the multicast source. OTV and VXLAN overlay tier IDs are unique:

```
BL1# show forwarding distribution tierpeerid otv

Tier-Peer-id allocations:
```

```

Type: Tier ID          Tier ID: 0x1  Tier Peer ID: 0x1000          Data: 0x0
Type: Tier Peer ID    Tier ID: 0x0  Tier Peer ID: 0x3fd  App: OTV  Data: 0x0
Type: Tier Peer ID    Tier ID: 0x1  Tier Peer ID: 0x1001 App: OTV  Data: 0x1
Type: Tier Peer ID    Tier ID: 0x1  Tier Peer ID: 0x1002 App: OTV  Data: 0x2
Type: Tier Peer ID    Tier ID: 0x1  Tier Peer ID: 0x13fd App: OTV  Data: 0x0

```

```
BL1# show forwarding distribution tierpeerid nve
```

```
Tier-Peer-id allocations:
```

```

Type: Tier ID          Tier ID: 0x2  Tier Peer ID: 0x2000          Data: 0x0

Type: Tier Peer ID    Tier ID: 0x2  Tier Peer ID: 0x2001 App: NVE Data:
0x1c1c1c1c Type: Tier Peer ID Tier ID: 0x2  Tier Peer ID: 0x2002 App:
NVE Data: 0x1b1b1b1b Type: Tier Peer ID  Tier ID: 0x2  Tier Peer ID:
0x23fd App: NVE  Data: 0x0

```

9.5.6 Troubleshooting VXLAN BGP EVPN and OTV Interoperation, and OTV with BDI

```

BL1# show tech-support arp
BL1# show tech-support otv
BL1# show tech-support lim
BL1# show tech-support oim
BL1# show tech-support nve
BL1# show tech-support vxlan-evpn

```

9.6 Feature History for VXLAN BGP EVPN and OTV Interoperation

This table lists the release history for this feature.

Table 8 : Feature History for VXLAN BGP EVPN and OTV interoperation

Feature Name	Release	Feature Information
VXLAN BGP EVPN and OTV interoperation	8.4(1)	<p>This feature was introduced to enable VXLAN and OTV (with BDI) configurations on a single switch (as a one box solution), and to enable OTV+BDI configurations on a single switch (as a one box solution).</p> <p>The following commands were introduced or modified for the OTV overlay:</p> <p>otv tunnel-stitch</p> <p>otv vni mapping</p> <p>show otv</p> <p>show otv internal overlay detail</p> <p>show otv internal tunnel-stitch</p> <p>show otv vni overlay</p> <p>The following command was introduced for the VXLAN overlay:</p> <p>tunnel-stitching enable</p> <p>The following command was introduced for the BDI function:</p> <p>ip address secondary anycast-primary</p>

CHAPTER 10 Proportional Multipath for VNF

This chapter contains the following sections:

- Information About Proportional Multipath for VNF.
- Licensing Requirements for Proportional Multipath for VNF.
- Guidelines and Limitations for Proportional Multipath for VNF.
- Default Setting for Proportional Multipath for VNF.
- Configuring Proportional Multipath for VNF.
- Verifying Proportional Multipath for VNF.
- Configuration Examples for Proportional Multipath for VNF.
- Additional References for Proportional Multipath for VNF.
- Feature History for Proportional Multipath for VNF.

10.1 Information About Proportional Multipath for VNF

In Network Function Virtualization Infrastructures (NFVi), anycast services networks are advertised from multiple Virtual Network Functions (VNFs). The Proportional Multipath for VNF feature enables advertising of all the available next hops to a given destination network. This feature enables the switch to consider all paths to a given route as equal cost multipath (ECMP) allowing the traffic to be forwarded using all the available links stretched across multiple ToRs.

As shown in Figure 1, North-South traffic that enters the VXLAN fabric at a border leaf is sent across all egress endpoints with the traffic forwarded proportional to the number of links from the egress Top-Of-Rack (TOR) to the destination network.

As shown in Figure 2, East-West traffic is forwarded between the VXLAN tunnel End Points (VTEPs) proportional to the number of next hops that are advertised by each Top-Of-Rack (TOR) switch to the destination network.

The switch uses BGP to advertise reachability within the fabric using the Layer 2 VPN (L2VPN)/Ethernet VPN (EVPN) address family. If all TOR switches and border leaves are within the same autonomous system (AS), a full internal BGP (iBGP) mesh is configured by using route reflectors or by having each BGP router peer with every other router.

Each TOR and border leaf constitutes a VTEP in the VXLAN fabric. You can use a BGP route reflector to reduce the full mesh BGP sessions across the VTEPs to a single BGP session between a VTEP and the route reflector. Virtual Network Identifiers (VNIs) are globally unique within the overlay. Each Virtual Routing and Forwarding (VRF) instance is mapped to a unique VNI. The inner destination MAC address in the VXLAN header belongs to the receiving VTEP that does the routing of the VXLAN payload. This MAC address is distributed as a BGP attribute along with the EVPN routes.

Multipath flattening is also supported. This leads to flattened IPv4 and IPv6 routes being sent to the FIB. The FIB then has a flattened set of next-hops that are sorted based on the increasing order of IP address.

The alternate deployment topology given below shows how a common loopback address is configured for the BGP connection between the ToR switches and the VNF instance.

10.1.1 Advertisement of Customer Networks

Customer networks are configured statically or learned locally by using Interior Gateway Protocol (IGP) or external BGP (eBGP) over a Provider Edge(PE)-Customer Edge(CE) link. In the data center fabric, the PE-CE link corresponds to access links from leaf devices to attached hosts (physical devices/VMs) or routers. These networks are redistributed into BGP and advertised to the VXLAN fabric.

The networks that are advertised to the TORs by the Virtual Machines (VMs) attached to them are advertised to

the VXLAN fabric as EVPN Type-5 routes with the following:

- The Route Distinguisher (RD) will be the L3VNI's configured RD.
- The gateway IP field will be populated with the next-hop.
- The next-hop of the EVPN route continues to be the VTEP-IP.
- The export route targets of the routes are derived from the configured export route targets of the associated Layer 3 VNI.

Multiple VRF routes may generate the same Type-5 Network Layer Reachability Information (NLRI) differentiated only by the gateway IP field as the routes are advertised with the L3VNI's RD and the gateway IP is not part of the Type-5 NLRI's key. The Network Layer Reachability Information (NLRI) is exchanged between BGP routers using UPDATE messages. These routes are advertised to the EVPN AF by extending the BGP export mechanism to include ECMPs and by using the add-path BGP feature in the EVPN AF.

Each Type-5 route within the EVPN AF that is created by using this feature may have multiple paths that are imported into the corresponding VRF based on the matching of the received route targets and by having ECMP enabled within the VRF and in the EVPN AF. Within the VRF, the route will be a single prefix with multiple paths. Each path represents a Type-5 EVPN path or those paths that are learned locally within the VRF. Those EVPN Type-5 routes that are enabled for this feature will have their next hop (NH) in the VRF derived from their gateway IP field. Use the **export-gateway-ip** command to enable BGP to advertise the gateway IP in the EVPN Type-5 routes.

Use the **maximum-paths mixed** command to enable BGP and the Unicast Routing Information Base (URIB) to consider the following paths as ECMP:

- iBGP paths
- eBGP paths
- Paths from other protocols (such as static) that are redistributed or injected into BGP.

The paths can be either local to the device (redistributed static or network-originated) or remote (eBGP or iBGP learned over BGP-EVPN or PE-CE). This overrides the default route selection behavior in which local routes are preferred over remote routes. URIB downloads all NHs of the route, including locally learned and user-configured routes, to the Unicast FIB Distribution Module (uFDM)/Forwarding Information Base (FIB).

10.1.2 Legacy Peer Support

Use the **advertise-gw-ip** command to advertise EVPN Type-5 routes with the gateway IP set. TORs will then advertise the gateway IP in the Type-5 NLRI. However, legacy peers running on INOS versions older than Inspur INOS Release 8.4(1) cannot process the Gateway IP which may lead to unexpected behavior. To prevent this scenario from occurring, use the **no advertise-gw-ip** command. BGP will then set the gateway IP field of the Type-5 NLRI to zero even if the path being advertised has a valid gateway IP.

The **no advertise-gw-ip** command flaps the specified peer session as gracefully as possible. The remote peer triggers graceful restart if the peer supports this capability. When the session is reestablished, the local peer advertises EVPN Type-5 routes with the gateway IP set or with the gateway IP as zero depending on whether the **advertise-gw-ip** command has been used or not. By default, this knob is enabled and the gateway IP field is populated with the appropriate next hop value.

10.2 Licensing Requirements for Proportional Multipath for VNF

The following table shows the licensing requirements for this feature:

Product	License Requirement
Inspur INOS	Proportional Multipath for VNF requires no license. Any feature not included in a license package is bundled with the Inspur INOS system images and is provided at no extra charge to you. For a complete explanation of the Inspur INOS licensing scheme, see the License and Copyright Information for Inspur INOS Software.

10.3 Guidelines and Limitations for Proportional Multipath for VNF

- Static and direct routes have to be redistributed into the BGP when the Proportional Multipath for VNF feature is enabled.

- Routes cannot be redistributed into BGP if OSPF or EIGRP is being used as an IGP.

- If the Proportional Multipath for VNF feature is enabled and the routes are not redistributed into BGP, asymmetric load balancing of traffic may occur as the local routes from URIB may not show up in BGP and on remote TORs as EVPN paths.

- Devices on which mixed-multipath is enabled must support the same load-balancing algorithm. Otherwise, traffic tromboning may occur.

- If a VNF instance is multi-homed to multiple TORs, policies have to be configured or BGP routes have to be originated using a network command. This results in each TOR's connection to the VNF being displayed in the BGP routing table. Each TOR can now see the VNF's direct routes to the other TORs in which the VNF is multi-homed. This allows each TOR to advertise paths to the Gateway IPs through other TORs leading to a next hop resolution loop.

Consider a scenario in which a VNF is multi-homed to two TORs, TOR1 and TOR2. Individual links to the TORs are addressed as 1.1.1.1 and 2.2.2.2. If the VNF advertises a service 192.168.1.0/24 through the TORs, the TORs advertise EVPN routes to 192.168.1.0/24 with Gateway IPs of 1.1.1.1 and 2.2.2.2 respectively.

This causes an issue with the Recursive Next Hop (RNH) resolution on a remote TOR (for example, TOR3). The gateway IP is resolved to a /24 route pointing to another gateway IP. That second gateway IP is resolved by a route pointing to the first gateway IP. So, in our scenario, the gateway IP 1.1.1.1 is resolved by 1.1.1.0/24 which points to 2.2.2.2. And 2.2.2.2 is resolved by 2.2.2.0/24 which points to 1.1.1.1.

The above condition occurs as both TORs connected to the VNF are advertising the VNF's connected routes. TOR1 is advertising 1.1.1.0/24 and 2.2.2.0/24. However, 1.1.1.0 is advertised without a gateway IP as it is a connected subnet on TOR1. Also, 2.2.2.0 is an OSPF route pointing to 1.1.1.1 which is the VNF's address connected to TOR1.

Similarly, TOR2 advertises both subnets and 2.2.2.0/24 is sent without a gateway IP as it is directly connected to TOR2. 1.1.1.0 is learned via OSPF and is sent with a gateway IP of 2.2.2.2 which is the VNF's address connected to TOR2. 1.1.1.1/32 and 2.2.2.2/32 will not be advertised as they are Adjacency Manager (AM) routes on each TOR.

This issue does not have a resolution when Type-5 routes are involved. However, this scenario can be avoided if the TORs advertise the gateway IP's /32 address using a network command. And if the gateway IPs are being resolved by Type-2 EVPN MAC/IP routes, this scenario can be avoided as the gateway IP will be resolved by the /32 IP route.

10.4 Default Setting for Proportional Multipath for VNF

Parameter	Default
Proportional Multipath for VNF	Disabled

10.5 Configuring Proportional Multipath for VNF

10.5.1 Configuring the Route Reflector

Procedure

- Step 1** Enter global configuration mode:
switch# **configure terminal**
- Step 2** Assign an autonomous system (AS) number to a device and enter the BGP configuration mode:
switch(config)# **router bgp** <500000>
- Step 3** Required: Configure Layer-2 VPN EVPN parameters in the VXLAN EVPN fabric:
switch(config-router)# **address-family l2vpn evpn**
- Step 4** Configure the capability of sending and receiving additional paths to and from all the neighbors in this address family:
switch(config-router-af)# **additional paths send**
switch(config-router-af)# **additional paths receive**
switch(config-router-af)# **additional paths selection route-map** <passall>
- Step 5** Exit the address family configuration mode:
switch(config-router-af)# **exit**
- Step 6** Exit the BGP configuration mode:
switch(config-router)# **exit**
- Step 7** Enter the route-map configuration mode and define the conditions for redistributing routes from one routing protocol to another:
switch(config)# **route-map** <passall> **permit** <10>
- Step 8** Set the path selection criteria:
switch(config-route-map)# **set path-selection all advertise**

Running Configuration

This example shows a running configuration for a route reflector device.

```
configure
  terminal
  router bgp
  500000
  address-family
  l2vpn evpn
  additional
  paths send
  additional
  paths receive
  additional paths selection
route-map passall route-map passall
permit 10
set path-selection all advertise
```


10.5.2 Configuring the ToR

Procedure

- Step 1** Enter global configuration mode:
switch# **configure terminal**
- Step 2** Configure BGP:
switch(config)# **router bgp** <500000>
- Step 3** Required: Configure address family Layer 2 VPN EVPN under router bgp context:
switch(config-router)# **address-family l2vpn evpn**
- Step 4** Enables BGP and the Unicast Routing Information Base (URIB) to consider the following paths as Equal Cost Multi Path (ECMP):
- iBGP paths
 - eBGP paths
 - Paths from other protocols (such as static) that are redistributed or injected into BGP
- switch(config-router-af)# **maximum-paths mixed** <32>
- Step 5** The additional-paths configuration for sending:
switch(config-router-af)# **additional paths send**
- Step 6** The additional-paths configuration for receiving:
switch(config-router-af)# **additional paths receive**
- Step 7** The additional-paths configuration applied for the route-map:
switch(config-router-af)# **additional paths selection route-map** <passall>
- Step 8** Exits the command mode:
switch(config-router-af)# **exit**
- Step 9** Enter the VRF configuration mode:
switch(config-router)# **vrf evpn-tenant-1001**
- Step 10** Configure address family for IPv4.
switch(config-router-vrf)# **address-family ipv4 unicast**
- Step 11** Enable BGP to advertise the gateway IP in EVPN Type-5 routes:
switch(config-router-vrf-af)# **export-gateway-ip**
- Step 12** Enables BGP and the Unicast Routing Information Base (URIB) to consider the following paths as Equal Cost Multi Path (ECMP):
- iBGP paths
 - eBGP paths
 - Paths from other protocols (such as static) that are redistributed or injected into BGP
- switch(config-router-vrf-af)# **maximum-paths mixed** <32>
- Step 13** Exits the command mode:
switch(config-router-vrf-af)# **exit**
- Step 14** Enter the address family configuration mode:

```
switch(config-router-vrf)# address-family ipv6 unicast
```

Step 15 Enable BGP to advertise the gateway IP in EVPN Type-5 routes:

```
switch(config-router-vrf-af)# export-gateway-ip
```

Step 16 Enables BGP and the Unicast Routing Information Base (URIB) to consider the following paths as Equal Cost Multi Path (ECMP):

- iBGP paths
- eBGP paths
- Paths from other protocols (such as static) that are redistributed or injected into BGP

```
switch(config-router-vrf-af)# maximum-paths mixed <32>
```

Step 17 Exits the comand mode:

```
switch(config-router-vrf-af)# exit
```

Step 18 Configure the route-map:

```
switch(config)# route-map passall permit <10>
```

Step 19 Sets the route-map related to the additional-paths feature: switch(config-route-map)# set path-selection all advertise

Running Configuration

This example shows a running configuration for a ToR device.

```
configure
 terminal
 router bgp
 500000
 address-family
 l2vpn evpn
 maximum-
 paths mixed
 32
 additional
 paths send
 additional
 paths
 receive
 additional paths selection route-
 map passall vrf cust_1
 address-
 family
 l2vpn evpn
 export-
 gateway-ip
 maximum-
 paths mixed
 32
 route-map passall permit 10
 set path-selection all advertise
```

10.5.3 Configuring the Border Leaf

Procedure

- Step 1** Enter global configuration mode:
switch# **configure terminal**
- Step 2** Configure BGP:
switch(config)# **router bgp** <500000>
- Step 3** Required: Configure address family Layer 2 VPN EVPN under router bgp context:
switch(config-router)# **address-family l2vpn evpn**
- Step 4** Enables BGP and the Unicast Routing Information Base (URIB) to consider the following paths as Equal Cost Multi Path (ECMP):
- iBGP paths
 - eBGP paths
 - Paths from other protocols (such as static) that are redistributed or injected into BGP.
- switch(config-router-af)# **maximum-paths mixed** <32>
- Step 5** The additional-paths configuration for sending:
switch(config-router-af)# **additional paths send**
- Step 6** The additional-paths configuration for receiving:
switch(config-router-af)# **additional paths receive**
- Step 7** The additional-paths configuration enables the additional-paths feature:
switch(config-router-af)# **additional paths selection route-map** <passall>
- Step 8** Exits the command mode:
switch(config-router-af)# **exit**
- Step 9** Enter the VRF configuration mode:
switch(config-router)# **vrf evpn-tenant-1001**
- Step 10** Configure address family for IPv4:
switch(config-router-vrf)# **address-family ipv4 unicast**
- Step 11** Enable BGP to advertise the gateway IP in EVPN Type-5 routes:
switch(config-router-vrf-af)# **export-gateway-ip**
- Step 12** Enables BGP and the Unicast Routing Information Base (URIB) to consider the following paths as Equal Cost Multi Path (ECMP):
- iBGP paths
 - eBGP paths
 - Paths from other protocols (such as static) that are redistributed or injected into BGP.
- switch(config-router-vrf-af)# **maximum-paths mixed** <32>
- Step 13** Exit the address family configuration mode:
switch(config-router-vrf-af)# **exit**

- Step 14** Enter the address family configuration mode:
switch(config-router-vrf)# **address-family ipv6 unicast**
- Step 15** Enable BGP to advertise the gateway IP in EVPN Type-5 routes:
switch(config-router-vrf-af)# **export-gateway-ip**
- Step 16** Enables BGP and the Unicast Routing Information Base (URIB) to consider the following paths as Equal Cost Multi Path (ECMP):
- iBGP paths
 - eBGP paths
 - Paths from other protocols (such as static) that are redistributed or injected into BGP.
- switch(config-router-vrf-af)# **maximum-paths mixed <32>**
- Step 17** Exits the command mode:
switch(config-router-vrf-af)# **exit**
- Step 18** Configure the route-map:
switch(config)# **route-map passall permit <10>**
- Step 19** Sets the route-map related to the additional-paths feature:
switch(config-route-map)# **set path-selection all advertise**
- Step 20** Exit the route-map configuration mode.
switch(config-route-map)# **exit**
- Step 21** Configure the unicast FIB load sharing algorithm for data traffic.
switch(config)# **ip load-sharing address source-destination rotate <32> universal-id <1>**

Note

- The **universal-id** option sets the random seed for the hash algorithm and shifts the flow from one link to another. You do not need to configure the universal ID. Inspur INOS chooses the Universal ID if you do not configure it. The range is from 1 to 4294967295.
- The **rotate** option causes the hash algorithm to rotate the link picking selection so that it does not continually choose the same link across all nodes in the network. It does so by influencing the bit pattern for the hash algorithm. This option shifts the flow from one link to another and load balances the already load-balanced (polarized) traffic from the first ECMP level across multiple links.
 - If you specify a **rotate** value, the 64-bit stream is interpreted starting from that bit position in a cyclic rotation. The **rotate** range is from 1 to 63, and the default is 32.
 - With multi-tier Layer 3 topology, polarization is possible. To avoid polarization, use a different rotate bit at each tier of the topology.
 - To configure a rotation value for port channels, use the **port-channel load-balance src-dst ip-l4port rotate rotate** command.

Running Configuration

This example shows a running configuration for a Border Leaf.

```
configure
terminal
router    bgp
500000
address-family
```

```

l2vpn    evpn
maximum-
paths    mixed
32
additional
paths    send
additional
paths    receive
additional paths selection route-
map passall vrf cust_1
address-
family
l2vpn    evpn
export-
gateway-ip
maximum-
paths    mixed
32
route-map passall permit 10
set path-selection all advertise
ip load-sharing address source-destination rotate 32 universal-id 1

```

10.5.4 Configuring a BGP Legacy Peer

Procedure

- Step 1 Enter global configuration mode:
switch# **configure terminal**
- Step 2 Assign an autonomous system (AS) number to a device and enter the BGP configuration mode:
switch(config)# **router bgp** <500000>
- Step 3 Configure a BGP neighbor and enter the neighbor configuration mode:
switch(config-router)# **neighbor** <102.102.102.102>
- Step 4 Specify autonomous system number of the neighbor:
switch(config-router-neighbor)# **remote-as** <2000000>
- Step 5 Enter the address family configuration mode:
switch(config-router-neighbor)# **address-family l2vpn evpn**
- Step 6 Disable the Proportional Multipath for VNF feature for legacy peers:
switch(config-router-neighbor-af)# **no advertise-gw-ip**

Note

EVPN Type-5 routes will now be sent with 0 as the gateway IP.

Running Configuration

This example shows a running configuration for a route reflector device.

```

router bgp 500000
neighbor
102.102.102.102
remote-as
2000000 address-
family    l2vpn

```

```

evpn
no advertise-gw-
ip

```

10.6 Verifying Proportional Multipath for VNF

Command	Purpose
show bgp ipv4 unicast	Displays Border Gateway Protocol (BGP) information for the IPv4 unicast address family.
show bgp l2vpn evpn	Displays BGP information for the Layer-2 Virtual Private Network (L2VPN) Ethernet Virtual Private Network (EVPN) address family.
show ip route	Displays routes from the unicast RIB.

The following example shows how to display BGP information for the L2VPN EVPN address family:

```

switch# show bgp l2vpn evpn 11.1.1.0
BGP routing table information for VRF default, address
family L2VPN EVPN Route Distinguisher: 13.13.13.13:3 //
Remote route
BGP routing table entry for [5]:[0]:[0]:[24]:[11.1.1.0]/224,
version 1341 Paths: (3 available, best #1)
Flags: (0x000002) on xmit-list, is not in l2rib/evpn,
is not in HW Multipath: eBGP

Advertised path-id 1
Path type: external, path is valid,
is best path Imported to 2
destination(s)
Gateway IP: 11.1.1.133
AS-Path: 2000000 100000 , path sourced external to
AS 11.11.11.11 (metric 5) from 102.102.102.102
(102.102.102.102)
Origin incomplete, MED not set, localpref
100, weight 0 Received label 22001
Received path-id 3
Extcommunity: RT:23456:22001 Route-
Import:11.11.11.11:2001 ENCAP:8 Router
MAC:003a.7d7d.1dbd

Path type: external, path is valid, not best reason: Neighbor
Address, multipath Imported to 2 destination(s)
Gateway IP: 11.1.1.233
AS-Path: 2000000 100 , path sourced
external to AS
33.33.33.33 (metric 5) from 102.102.102.102 (102.102.102.102)
Origin incomplete, MED not set, localpref
100, weight 0 Received label 22001
Received path-id 2
Extcommunity: RT:23456:22001 Route-
Import:33.33.33.33:2001 ENCAP:8 Router
MAC:e00e.da4a.589d

Path type: external, path is valid, not best reason: Neighbor
Address, multipath Imported to 2 destination(s)

```

```

Gateway IP: 11.1.1.100
AS-Path: 2000000 500000 , path sourced external to
AS 22.22.22.22 (metric 5) from 102.102.102.102
(102.102.102.102)
Origin incomplete, MED not set, localpref
100, weight 0 Received label 22001
Received path-id 1
Extcommunity: RT:23456:22001 Route-
Import:22.22.22.22:2001 ENCAP:8 Router
MAC:e00e.da4a.62a5

Path-id 1 not advertised to any peer

Route Distinguisher: 4.4.4.4:3 (L3VNI 22001) //
Local L3VNI BGP routing
table entry for [5]:[0]:[0]:[24]:[11.1.1.0]/224, version 3465
Paths: (3 available, best #1)
Flags: (0x000002) on xmit-list, is not in l2rib/evpn,
is not in HW Multipath: eBGP

Advertised path-id 1
Path type: external, path is valid, is best
path

Imported from
13.13.13.13:3:[5]:[0]:[0]:[24]:[11.1.1.0]/224 Gateway IP:
11.1.1.100
AS-Path: 2000000 500000 , path sourced external to AS
22.22.22.22 (metric 5) from 102.102.102.102
(102.102.102.102)
Origin incomplete, MED not set, localpref 100,
weight 0 Received label 22001
Received path-id 1
Extcommunity: RT:23456:22001 Route-Import:22.22.22.22:2001
ENCAP:8 Router MAC:e00e.da4a.62a5

Path type: external, path is valid, not best reason: newer EBGP path, multipat
h

Imported from
13.13.13.13:3:[5]:[0]:[0]:[24]:[11.1.1.0]/224 Gateway IP:
11.1.1.233
AS-Path: 2000000 100 , path sourced external to AS
33.33.33.33 (metric 5) from 102.102.102.102 (102.102.102.102)
Origin incomplete, MED not set, localpref 100,
weight 0 Received label 22001
Received path-id 2
Extcommunity: RT:23456:22001 Route-Import:33.33.33.33:2001
ENCAP:8 Router MAC:e00e.da4a.589d

Path type: external, path is valid, not best reason: newer EBGP path, multipat
h

Imported from
13.13.13.13:3:[5]:[0]:[0]:[24]:[11.1.1.0]/224 Gateway IP:
11.1.1.133
AS-Path: 2000000 100000 , path sourced external to AS
11.11.11.11 (metric 5) from 102.102.102.102
(102.102.102.102)
Origin incomplete, MED not set, localpref 100,
weight 0 Received label 22001
Received path-id 3
Extcommunity: RT:23456:22001 Route-Import:11.11.11.11:2001
ENCAP:8 Router MAC:003a.7d7d.1dbd

```

Path-id 1 not advertised to any peer

The following example shows how to display BGP information for the IPv4 unicast address family:

```
switch# show bgp ipv4 unicast 11.1.1.0 vrf cust_1
BGP routing table information for VRF cust_1, address family IPv4
Unicast BGP routing table entry for 11.1.1.0/24, version 4
Paths: (3 available, best #1)
Flags: (0x80080012) on xmit-list, is in urib, is backup urib route,
is in HW vpn: version 1093, (0x100002) on xmit-list
Multipath: eBGP iBGP

Advertised path-id 1, VPN AF advertised path-id 1
Path type: external, path is valid, is best path, in rib
Imported from 13.13.13.13:3:[5]:[0]:[0]:[24]:[11.1.1.0]/224
AS-Path: 2000000 500000 , path sourced external to AS
11.1.1.100 (metric 5) from 102.102.102.102
(102.102.102.102)
Origin incomplete, MED not set, localpref 100,
weight 0 Received label 22001
Received path-id 1
Extcommunity: RT:23456:22001 Route-Import:22.22.22.22:2001
ENCAP:8 Router MAC:e00e.da4a.62a5

Path type: external, path is valid, not best reason: Neighbor Address, multipath,
in rib Imported from 13.13.13.13:3:[5]:[0]:[0]:[24]:[11.1.1.0]/224
AS-Path: 2000000 100 , path sourced external to AS
11.1.1.233 (metric 5) from 102.102.102.102 (102.102.102.102)
Origin incomplete, MED not set, localpref 100,
weight 0 Received label 22001
Received path-id 2

Extcommunity: RT:23456:22001 Route-
Import:33.33.33.33:2001 ENCAP:8 Router
MAC:e00e.da4a.589d

Path type: external, path is valid, not best reason: Neighbor Address,
multipath, in rib Imported from
13.13.13.13:3:[5]:[0]:[0]:[24]:[11.1.1.0]/224
AS-Path: 2000000 100000 , path sourced external to
AS 11.1.1.133 (metric 5) from 102.102.102.102
(102.102.102.102)
Origin incomplete, MED not set, localpref
100, weight 0 Received label 22001
Received path-id 3
Extcommunity: RT:23456:22001 Route-
Import:11.11.11.11:2001 ENCAP:8 Router
MAC:003a.7d7d.1dbd

VRF advertise information:
Path-id 1 not advertised to any peer

VPN AF advertise information:
Path-id 1 not advertised to any peer
```

The following example shows how to display routes from the unicast RIB after the Proportional Multipath for VNF feature has been configured:

```
switch# show ip route 1.1.1.0
vrf cust_1 IP Route Table for
VRF "cust_1"
```



```

...
1.1.1.0/24,   ubest/mbest:  22/0,   all-best
(0x300003d)
  *via  3.0.0.1,   [1/0],
        08:13:17,   static
        recursive next hop:
        3.0.0.1/32
  *via  3.0.0.2,   [1/0],
        08:13:17,   static
        recursive next hop:
        3.0.0.2/32
  *via  3.0.0.3,   [1/0],
        08:13:16,   static
        recursive next hop:
        3.0.0.3/32
  *via  3.0.0.4,   [1/0],
        08:13:16,   static
        recursive next hop:
        3.0.0.4/32
  *via  2.0.0.1, [200/0], 06:09:19, bgp-2, internal, tag 2 (evpn) segid:
3003802 tunnelid: 0x300003e encap: VXLAN
        BGP-EVPN: VNI=3003802 (EVPN)
        client-specific
        data: 3b recursive
        next          hop:
        2.0.0.1/32
        extended route information: BGP origin AS 2 BGP peer AS 2
  *via  2.0.0.2, [200/0], 06:09:19, bgp-2, internal, tag 2 (evpn) segid:
3003802 tunnelid: 0x300003e encap: VXLAN
        BGP-EVPN: VNI=3003802 (EVPN)
        client-specific
        data: 3b recursive
        next          hop:
        2.0.0.2/32
        extended route information: BGP origin AS 2 BGP peer AS 2

```

Refer the topologies in the figures given below for the sample outputs in this section.

The following example shows how to display the common loopback path for IPv4 addressing scenarios:

```

CN-12700-BL3# show ip route 104.1.1.0 vrf
os-vrf100 104.1.1.0/24, ubest/mbest: 1/0
time, all-best (0xb011801)
*via 192.1.34.1, [200/0], 12:09:06, bgp-65000, internal, tag 99 (evpn),
segid: 10002 tunnelid: 0xb010301 encap: VXLAN

```

The following example shows how to display the total number of recursive next hop (RNH) ECMP paths that point to the gateway IPv4 address in the unicast routing information base (URIB):

```

CN-12700-BL3# show ip route 192.1.34.1 vrf
os-vrf100 192.1.34.1/32, ubest/mbest: 8/0
time, all-best (0xb011801)
*via 203.1.3.2, [200/0], 12:11:39, bgp-65000, internal, tag 65000
(evpn), segid: 10002 tunnelid: 0xb010301 encap: VXLAN
*via 203.1.4.2, [200/0], 12:11:39, bgp-65000, internal, tag 65000
(evpn), segid: 10002 tunnelid: 0xb010301 encap: VXLAN
*via 204.1.5.2, [200/0], 12:11:39, bgp-65000, internal, tag 65000
(evpn), segid: 10002 tunnelid: 0xb010401 encap: VXLAN
*via 204.1.6.2, [200/0], 12:11:39, bgp-65000, internal, tag 65000 (evpn),
segid:
10002 tunnelid: 0xb010401 encap: VXLAN
*via 204.1.7.2, [200/0], 12:11:39, bgp-65000, internal, tag 65000 (evpn),

```

```

segid: 10002 tunnelid: 0xb010401 encap: VXLAN
*via 204.1.8.2, [200/0], 12:11:39, bgp-65000, internal, tag 65000 (evpn),
segid: 10002 tunnelid: 0xb010401 encap: VXLAN
*via 99.1.16.1, [200/0], 12:11:39, bgp-65000, internal, tag 65000 (evpn),
segid: 10002 tunnelid: 0xb011001 encap: VXLAN
*via 99.1.17.1, [200/0], 12:11:39, bgp-65000, internal, tag 65000 (evpn),
segid: 10002 tunnelid: 0xb011101 encap: VXLAN

```

The following example shows how to display the total number of RNH ECMP paths that point to the gateway IPv4 address and are flattened in the unicast forwarding information base (UFIB):

```

CN-12700-BL3# show forwarding route 104.1.1.0 vrf os-vrf100
slot 9
=====

IPv4 routes for table os-
vrf100/base '*' denotes
recursive route
-----+-----+-----+-----+-----+-----+-----+-----+-----+
Prefix          | Next-hop                               | Interface          |
Labels          |                                         |                   |
-----+-----+-----+-----+-----+-----+-----+-----+-----+
104.1.1.0/24    | 11.1.3.1                               | nve1               | (tunnel 0xb010301)
                | 11.1.3.1                               | nve1               | (tunnel 0xb010301)
                | 11.1.4.1                               | nve1               | (tunnel 0xb010401)
                | 11.1.4.1                               | nve1               | (tunnel 0xb010401)
                | 11.1.4.1                               | nve1               | (tunnel 0xb010401)
                | 11.1.4.1                               | nve1               | (tunnel 0xb010401)
                | 11.1.16.1                              | nve1               | (tunnel 0xb011001)
                | 11.1.17.1                              | nve1               | (tunnel 0xb011101)

```

The following example shows how to display the common loopback path for IPv6 addressing scenarios:

```

CN-12700-BL3# show ipv6 route 104:1:1:1:: vrf os-
vrf100 104:1:1:1::/64, ubest/mbest: 1/0, all-best
(Oxb011801)
*via 192:1:34::1/128, [200/0], 12:19:00, bgp-65000, internal, tag 99
(evpn), segid 10002 tunnel: 0xb010301 encap: VXLAN

```

The following example shows how to display the total number of RNH ECMP paths that point to the gateway IPv6 address in the URIB:

```

CN-12700-BL3# show ipv6 route 192:1:34::1 vrf os-
vrf100 192:1:34::1/128, ubest/mbest: 8/0, all-
best (Oxb011801)
*via 203:1:3::2/128, [200/0], 12:20:06, bgp-65000, internal, tag 65000
(evpn), segid 10002 tunnel: 0xb010301 encap: VXLAN
*via 203:1:4::2/128, [200/0], 12:20:06, bgp-65000, internal, tag 65000
(evpn), segid 10002 tunnel: 0xb010301 encap: VXLAN
*via 204:1:5::2/128, [200/0], 12:40:59, bgp-65000, internal, tag 65000
(evpn), segid 10002 tunnel: 0xb010401 encap: VXLAN
*via 204:1:6::2/128, [200/0], 12:40:59, bgp-65000, internal, tag 65000
(evpn), segid 10002 tunnel: 0xb010401 encap: VXLAN
*via 204:1:7::2/128, [200/0], 12:40:59, bgp-65000, internal, tag 65000
(evpn), segid 10002 tunnel: 0xb010401 encap: VXLAN
*via 204:1:8::2/128, [200/0], 12:40:59, bgp-65000, internal, tag 65000
(evpn), segid 10002 tunnel: 0xb010401 encap: VXLAN
*via 99:1:16::1/128, [200/0], 12:40:38, bgp-65000, internal, tag 65000
(evpn), segid 10002 tunnel: 0xb011001 encap: VXLAN
*via 99:1:17::1/128, [200/0], 12:40:45, bgp-65000, internal, tag 65000

```



```

Origin IGP, MED not set, localpref
100, weight 0 Received label 10002
Received path-id 1
Extcommunity: RT:65000:51300 ENCAP:8 Router
MAC:2cd0.2d56.931f Originator: 11.0.3.1 Cluster
list: 10.1.1.1

```

The following example shows how to display the IPv6 BGP EVPN route type5 information in a VRF. The sample output shows that the multipath mode is mixed along with the Gateway IPv6 address that is used.

```

CN-12700-BL3# show bgp l2vpn evpn 104:1:1:1:
BGP routing table information for VRF default, address
family L2VPN EVPN Route Distinguisher: 11.0.3.1:29
BGP routing table entry for [5]:[0]:[0]:[64]:[104:1:1:1:]/416,
version 1712190 Paths: (2 available, best #2)
Flags: (0x000002) (high32 00000000) on xmit-list, is not in l2rib/evpn, is
not in HW Multipath: Mixed

Path type: internal, path is valid, not best reason: Neighbor Address, no labeled

nexthop Gateway IP: 192:1:34::1
AS-Path: 99 98 , path sourced external to AS
11.1.3.1 (metric 3) from 10.1.2.1 (10.1.2.1)
Origin IGP, MED not set, localpref 100,
weight 0 Received label 10002
Received path-id 1
Extcommunity: RT:65000:51300 ENCAP:8 Router
MAC:2cd0.2d56.931f Originator: 11.0.3.1 Cluster list:
10.1.2.1

Advertised path-id 1
Path type: internal, path is valid, is best path, no labeled
nexthop Imported to 2 destination(s)
Imported paths list: issu300 default

Gateway IP: 192:1:34::1
AS-Path: 99 98 , path sourced external
to AS 11.1.3.1 (metric 3) from
10.1.1.1 (10.1.1.1)
Origin IGP, MED not set, localpref 100,
weight 0 Received label 10002
Received path-id 1
Extcommunity: RT:65000:51300 ENCAP:8 Router
MAC:2cd0.2d56.931f Originator: 11.0.3.1 Cluster list:
10.1.1.1

```

The following example displays IPv4 Multipath hashing information. This will display the route selected for a particular source and destination address along with the VTEP and the egress interface.

```

CN-12700-BL3# show routing hash 200.1.1.1 104.1.1.1 vrf os-
vrf100 Load-share parameters used for software forwarding:
load-share mode: address source-
destination Universal-id seed:
0xffffffff
Hash for VRF
"os-vrf100" Hash
Type is 1
Hashing to path *11.1.4.1
Eth10/24%default MPLS[2]:
Label=10002 E=0 TTL=0 S=0
MPLS[1]: Label=10002 E=0 TTL=0
S=0 MPLS[0]: Label=10002 E=0
TTL=0 S=0
For route:
104.1.1.0/24, ubest/mbest: 1/0 time, all-best (0xb011801)
*via 192.1.34.1, [200/0], 12:41:23, bgp-65000, internal, tag 99 (evpn),

```

```
segid: 10002 tunnelid: 0xb010301 encap: VXLAN
```

The following example displays IPv6 Multipath hashing information. This will display the route selected for a particular source and destination address along with the VTEP and the egress interface.

```
CN-12700-BL3# show routing ipv6 hash 200:1:1:1::1 104:1:1:1::1 vrf os-
vrf100 Load-share parameters used for software forwarding:
load-share mode: address source-
destination Universal-id seed:
0xffffffff
No IP protocol specified,
defaulting to UDP Hash for VRF "os-
vrf100"
Hash Type is 1
Hashing to path *:ffff:11.1.14.1
Ethernet10/24 MPLS[1]: Label=10002
E=0 TTL=0 S=0
MPLS[0]: Label=10002 E=0 TTL=0 S=0
For route:
104:1:1:1::/64, ubest/mbest: 26/0, all-best (0xb011801)
*via 192:1:34::1/128, [200/0], 00:01:32, bgp-65000, internal, tag 99 (evpn),
segid 10002 tunnel: 0xb010301 encap: VXLAN
```

The following example shows how to display the common loopback path for IPv4 addressing scenarios:

```
ToR-Leaf-Switch-3# show ip route 104.1.1.0
vrf os-vrf100 104.1.1.0/24, ubest/mbest: 1/0,
all-best (0xb010301)
*via 192.1.34.1, [20/0], 13:24:29, bgp-65000, external, tag 99
```

The following example shows how to display the total number of local and remote RNH ECMP paths that point to the gateway IPv4 address in the URIB:

```
ToR-Leaf-Switch-3# show ip route 192.1.34.1
vrf os-vrf100 192.1.34.1/32, ubest/mbest: 8/0,
all-best (0xb010301)
*via 203.1.3.2, Vlan3, [1/0], 14:46:05,
static
*via 203.1.4.2, Vlan4, [1/0], 14:46:04,
static
*via 204.1.5.2, [200/0], 01:53:23, bgp-65000, internal, tag 65000 (evpn)
segid: 10002 tunnelid: 0xb010401 encap: VXLAN

*via 204.1.6.2, [200/0], 01:52:08, bgp-65000, internal, tag 65000 (evpn)
segid: 10002 tunnelid: 0xb010401 encap: VXLAN

*via 204.1.7.2, [200/0], 01:53:23, bgp-65000, internal, tag 65000 (evpn)
segid: 10002 tunnelid: 0xb010401 encap: VXLAN

*via 204.1.8.2, [200/0], 01:52:08, bgp-65000, internal, tag 65000 (evpn)
segid: 10002 tunnelid: 0xb010401 encap: VXLAN

*via 99.1.16.1, [200/0], 14:45:14, bgp-65000, internal, tag 65000 (evpn)
segid: 10002 tunnelid: 0xb011001 encap: VXLAN

*via 99.1.17.1, [200/0], 14:45:14, bgp-65000, internal, tag 65000 (evpn)
segid: 10002 tunnelid: 0xb011101 encap: VXLAN
```

The following example shows how to display the total number of local and remote RNH ECMP paths that point to the gateway IPv4 address in the UFIB:

```
ToR-Leaf-Switch-3# show forwarding route 104.1.1.0 vrf os-vrf100

slot 1
=====
```

```
IPv4 routes for table os-vrf100/base
-----|-----|-----|-----|
Prefix          | Next-hop                                | Interface          | Labels
| Partial Install
-----|-----|-----|-----|
104.1.1.0/24
          203.1.3.2                        Vlan3
          203.1.4.2                        Vlan4
          11.1.4.1                          nve1
          11.1.4.1                          nve1
          11.1.4.1                          nve1
          11.1.4.1                          nve1
          11.1.16.1                         nve1
          11.1.17.1                         nve1
```

The following example shows how to display the common loopback path for IPv6 addressing scenarios:

```
ToR-Leaf-Switch-3# show ipv6 route 104:1:1:1::
vrf os-vrf100 104:1:1:1::/64, ubest/mbest: 1/0,
all-best (0xb010301)
*via 192:1:34::1/128, [200/0], 02:04:49, bgp-65000, external, tag 99
```

The following example shows how to display the total number of local and remote RNH ECMP paths that point to the gateway IPv6 address in the URIB:

```
ToR-Leaf-Switch-3# show ipv6 route 192:1:34::1
vrf os-vrf100 192:1:34::1/128, ubest/mbest: 8/0,
all-best (0xb010301)
*via 203:1:3::2, Vlan3, [1/0], 15:00:43, static
*via 203:1:4::2, Vlan4, [1/0], 15:00:42, static
*via 204:1:8::2/128, [200/0], 02:06:54, bgp-65000, internal, tag 65000
(evpn) segid 10002 tunnel: 0xb010401 encap: VXLAN
*via 204:1:7::2/128, [200/0], 02:06:54, bgp-65000, internal, tag 65000
(evpn) segid 10002 tunnel: 0xb010401 encap: VXLAN
*via 204:1:5::2/128, [200/0], 02:08:09, bgp-65000, internal, tag 65000
(evpn) segid 10002 tunnel: 0xb010401 encap: VXLAN
*via 204:1:6::2/128, [200/0], 02:08:09, bgp-65000, internal, tag 65000
(evpn) segid 10002 tunnel: 0xb010401 encap: VXLAN
*via 99:1:16::1/128, [200/0], 02:08:09, bgp-65000, internal, tag 65000
(evpn) segid 10002 tunnel: 0xb011001 encap: VXLAN
*via 99:1:17::1/128, [200/0], 02:08:09, bgp-65000, internal, tag 65000
(evpn) segid 10002 tunnel: 0xb011101 encap: VXLAN
```

The following example shows how to display the total number of local and remote RNH ECMP paths that point to the gateway IPv6 address in the U6FIB:

```
N93K-EX-LF3# show forwarding ipv6 route 104:1:1:1:: vrf os-vrf100
slot 1
=====
IPv6 routes for table os-vrf100/base
-----|-----|-----|-----|
Prefix          | Next-hop                                | Interface          | Labels
```

| Partial Install

```

----- 104:1:1:1::/64
                203:1:3::2          Vlan3
                203:1:4::2          Vlan4
                11.1.4.1            nve1
                11.1.4.1            nve1
                11.1.4.1            nve1
                11.1.4.1            nve1
                11.1.16.1           nve1
                11.1.17.1           nve1

```

The following example shows how to display the IPv4 BGP EVPN route type5 information in a VRF. The sample output shows that the multipath mode is mixed along with the Gateway IP address that is used.

```

ToR-Leaf-Switch-3# show bgp l2vpn evpn 104.1.1.0
BGP routing table information for VRF default, address family
L2VPN EVPN Route Distinguisher: 11.0.4.1:28
BGP routing table entry for [5]:[0]:[0]:[24]:[104.1.1.0]/224,
version 26579 Paths: (2 available, best #2)
Flags: (0x000002) (high32 00000000) on xmit-list, is not in l2rib/evpn, is
not in HW Multipath: Mixed

Path type: internal, path is valid, not best reason: Neighbor Address, no labeled

nexthop Gateway IP: 192.1.34.1
AS-Path: 99 98 , path sourced external to AS
 11.1.4.1 (metric 3) from 10.1.2.1 (10.1.2.1)
  Origin IGP, MED not set, localpref 100,
  weight 0 Received label 10002
  Received path-id 1

  Extcommunity:   RT:65000:51300   ENCAP:8   Router
  MAC:2cd0.2d56.918f Originator: 0.0.0.0 Cluster list:
  10.1.2.1

Advertised path-id 1
Path type: internal, path is valid, is best path, no
labeled nexthop Imported to 2 destination(s)
Imported paths list:
issu300 default Gateway IP:
192.1.34.1
AS-Path: 99 98 , path sourced
external to AS 11.1.4.1 (metric 3)
from 10.1.1.1 (10.1.1.1)
Origin IGP, MED not set, localpref
100, weight 0 Received label 10002
Received path-id 1
Extcommunity:   RT:65000:51300   ENCAP:8   Router
MAC:2cd0.2d56.918f Originator: 0.0.0.0 Cluster list:
10.1.1.1

```

The following example shows how to display the IPv6 BGP EVPN route type5 information in a VRF. The sample output shows that the multipath mode is mixed along with the Gateway IP address that is used.

```

ToR-Leaf-Switch-3# show bgp l2vpn evpn 104:1:1:1::
BGP routing table information for VRF default, address
family L2VPN EVPN Route Distinguisher: 11.0.4.1:28
BGP routing table entry for [5]:[0]:[0]:[64]:[104:1:1:1::]/416,
version 21947 Paths: (2 available, best #1)
Flags: (0x000002) (high32 00000000) on xmit-list, is not in l2rib/evpn,
is not in HW Multipath: Mixed

Advertised path-id 1
Path type: internal, path is valid, is best path, no

```

```

        labeled nexthop Imported to 2 destination(s)
        Imported paths list:
issu300 default Gateway IP:
192:1:34::1
AS-Path: 99 98 , path sourced
external to AS 11.1.4.1 (metric 3)
from 10.1.1.1 (10.1.1.1)
Origin IGP, MED not set, localpref
100, weight 0 Received label 51300
Received path-id 1
Extcommunity: RT:65000:51300 ENCAP:8 Router
MAC:2cd0.2d56.918f Originator: 0.0.0.0 Cluster list:
10.1.1.1

Path type: internal, path is valid, not best reason: Neighbor Address, no

labeled nexthop Gateway IP: 192:1:34::1
AS-Path: 99 98 , path sourced external to AS
11.1.4.1 (metric 3) from 10.1.2.1
(10.1.2.1)
Origin IGP, MED not set, localpref
100, weight 0 Received label 51300
Received path-id 1
Extcommunity: RT:65000:51300 ENCAP:8 Router
MAC:2cd0.2d56.918f Originator: 0.0.0.0 Cluster list:
10.1.2.1

```

The following example displays IPv4 Multipath hashing information. This will display the route selected for a particular source and destination address along with the VTEP and the egress interface. In this example, a local path is selected. This indicates that the border leaf and the access leaf have the same hash result.

```

ToR-Leaf-Switch-3# show routing hash 200.1.1.1 104.1.1.1
vrf os-vrf100 Load-share parameters used for software
forwarding:
load-share mode: address source-
destination gtpu-teid No IPv4 protocol
specified, defaulting to UDP
Hash for VRF
"os-vrf100"
Hashing to path
*203.1.3.2

Out Interface: Vlan3

MPLS[1]: Label=10002
E=0 TTL=0 S=0 MPLS[0]:
Label=10002 E=0 TTL=0
S=0

For route:
104.1.1.0/24, ubest/mbest: 1/0, all-best (0xb010301)
*via 192.1.34.1, [20/0], 15:39:16, bgp-65000, external, tag 99

```

The following example also displays IPv4 Multipath hashing information. In this example, a remote path is selected. This indicates that the border leaf and the access leaf have different hash results.

```

ToR-Leaf-Switch-LF17# show routing hash 200.1.2.1 104.1.2.1 vrf
os-vrf100 Load-share parameters used for software forwarding:
load-share mode: address source-destination
gtpu-teid No IPv4 protocol specified,
defaulting to UDP
Hash for VRF "os-
vrf100" Hashing to
path *11.1.16.1

Out Interface: Eth1/50

MPLS[1]: Label=10002 E=0

```



```

TTL=0      S=0      MPLS[0]:
Label=10002 E=0 TTL=0 S=0
For route:
104.1.2.0/24, ubest/mbest: 1/0, all-best (0xb011101)
  *via 192.1.34.1, [20/0], 15:54:05, bgp-65000, external, tag 99

```

Now, go to Leaf 16 and use the **show routing hash** command. The sample output is as given below.

```

ToR-Leaf-Switch-LF16# show routing hash 200.1.2.1 104.1.2.1 vrf
os-vrf100 Load-share parameters used for software forwarding:
load-share mode: address source-destination
gtpu-teid No IPv4 protocol specified,
defaulting to UDP
Hash for VRF "os-
vrf100" Hashing to
path *99.1.16.1
  Out Interface: Eth1/48

MPLS[1]: Label=10002 E=0
TTL=0      S=0      MPLS[0]:
Label=10002 E=0 TTL=0 S=0
For route:
104.1.2.1/24, ubest/mbest: 1/0, all-best (0xb011001)
  *via 192.1.34.1, [20/0], 15:58:13, bgp-65000, external, tag 99

```

The following example displays IPv6 Multipath hashing information. This will display the route selected for a particular source and destination address along with the VTEP and the egress interface. In this example, a local path is selected. This indicates that the border leaf and the access leaf have the same hash result.

```

ToR-Leaf-Switch-3# show routing ipv6 hash 200:1:1:1::1 104:1:1:1::1 vrf
os-vrf100 Load-share parameters used for software forwarding:
load-share mode: 6
No IP protocol specified, defaulting to UDP

Hash for VRF
"os-vrf100" Hash
Type is 3
Hashing to path
*203:1:3::2      Out
Interface Vlan3
For route:
104:1:1:1::/64, ubest/mbest: 1/0, all-best (0xb010301)
  *via 192:1:34::1/128, [20/0], 03:08:55, bgp-65000, external, tag 99

```

The following example also displays IPv6 Multipath hashing information. In this example, a remote path is selected. This indicates that the border leaf and the access leaf have different hash results.

```

ToR-Leaf-Switch-LF17# show routing ipv6 hash 200:1:1:2::1 104:1:1:2::1 vrf
os-vrf100 Load-share parameters used for software forwarding:
load-share mode: 6
No IP protocol specified, defaulting to UDP

Hash for VRF
"os-vrf100" Hash
Type is 3
Hashing to path *::ffff:11.1.3.1
Out      Interface
Ethernet1/50 For
route:
104:1:1:2::/64, ubest/mbest: 1/0, all-best
(0xb011101)
  *via 192:1:34::1/128, [20/0], 03:13:12, bgp-65000, external, tag 99

```

Now, go to Leaf 3 and use the **show routing ipv6 hash** command. The sample output is as given below.

```

ToR-Leaf-Switch-LF3# show routing ipv6 hash 200:1:1:2::1 104:1:1:2::1 vrf os-

```

```

vrf100

Load-share parameters used for software
forwarding: load-share mode: 6
No IP protocol specified, defaulting to UDP

Hash for VRF
"os-vrf100"
Hash Type is
3
Hashing to path
*203:1:3::2 Out
Interface Vlan3
For route:
104:1:1:2::/64, ubest/mbest: 1/0, all-best
(0xb010301)
    *via 192:1:34::1/128, [20/0], 03:19:09, bgp-65000, external, tag 99

```

10.7 Configuration Examples for Proportional Multipath for VNF

This example shows a running BGP TOR configuration to enable BGP peering with the spine in a different autonomous system.

```

vrf context cust_1
  vni 22001
rd auto
address-family ipv4
  unicast route-
  target import
  23456:22001
  route-target import
  23456:22001 evpn route-
  target export
  23456:22001 route-target
  export 23456:22001 evpn
  route-target both auto
  route-target both auto evpn

  ip route 200.1.1.1 via 10.1.1.1 tag 100
  ip route 200.1.1.1 via 10.1.2.1 tag 100

route-map REDIST
  10 permit 10
route-map passall
  permit 10
  set path-selection all advertise

router bgp 500000
address-family l2vpn evpn maximum-paths mixed
32 additional-paths send additional-paths
receive
additional-paths selection route-map passall

neighbor 102.102.102.102
remote-as 2000000 update-source loopback0
ebgp-multihop 255
address-family ipv4 unicast allowas-in 3
send-community extended
address-family l2vpn evpn
  allowas-in 3

  send-community

```

```

extended neighbor
105.105.105.105
  remote-as
  2000000 update-
  source
  loopback0 ebgp-
  multihop 255
  address-family ipv4
    unicast allowas-
    in 3
    send-community
  extended address-
  family l2vpn evpn
    allowas-in 3
    send-community
  extended no
  advertise-gw-ip

vrf cust_1
  address-family ipv4
    unicast advertise
    l2vpn evpn wait-
    igp-convergence
  redistribute direct route-
  map REDIST redistribute
  static route-map REDIST
  export-gateway-ip
  maximum-paths mixed 32

```

This example shows a running legacy peer configuration.

```

router bgp 2000000
  neighbor 4.4.4.4
    remote-as
    500000 update-
    source
    loopback0 ebgp-
    multihop 255
    address-family ipv4
      unicast disable-
      peer-as-check
      send-community extended

  address-family
    l2vpn evpn
    disable-peer-
    as-check send-
    community
    extended

  neighbor 6.6.6.6
    remote-as 100
    update-source
    loopback0 ebgp-
    multihop 255
    address-family ipv4
      unicast send-
      community
    extended address-
    family ipv4 mvpn
      disable-peer-
      as-check send-
      community
    extended
    route-map setnh_unchanged out

```

```

address-family
  l2vpn      evpn
  disable-peer-
  as-check  send-
  community
  extended
  route-map
  setnh_unchanged  out
  no advertise-gw-ip
    
```

10.8 Additional References for Proportional Multipath for VNF

This section describes additional information related to implementing Proportional Multipath for VNF.

Related Documents

Related Topic
Inspur INOS licensing
Verified Scalability

10.9 Feature History for Proportional Multipath for VNF

This table lists the release history for this feature.

Feature Name	Release	Information
Proportional Multipath for VNF	8.4(1)	In Network Function Virtualization Infrastructures (NFVi), anycast services networks are advertised from multiple Virtual Network Functions (VNFs). The Proportional Multipath for VNF feature enables advertising of all the available next hops to a given destination network. This feature enables the switch to consider all paths to a given route as equal cost multipath (ECMP) allowing the traffic to be forwarded using all the available links stretched across multiple ToRs.

CHAPTER 11 Centralized VRF Route-Leaking for VXLAN BGP EVPN Fabrics

VXLAN BGP EVPN uses MP-BGP and its route-policy concept to import and export prefixes. The ability of this very extensive route-policy model allows to leak routes from one VRF to another VRF and vice-versa; any combination of custom VRF or VRF default can be used. VRF route-leaking is a switch-local function at specific to a location in the network, the location where the cross-VRF route-target import/export configuration takes place (leaking point). The forwarding between the different VRFs follows the control-plane, the location of where the configuration for the route-leaking is performed - hence Centralized VRF

route-leaking. With the addition of VXLAN BGP EVPN, the leaking point requires to advertise the cross-VRF imported/exported route and advertise them towards the remote VTEPs or External Routers.

The advantage of Centralized VRF route-leaking is that only the VTEP acting as leaking point requires the special capabilities needed, while all other VTEPs in the network are neutral to this function.

You can also enable the VRFs in the EVPN fabric to access shared services in a specific VRF. These shared services are provided by a GOLF or Data Center Interconnect (DCI) device. The hosts located behind the internet or core can access the hosts inside the Application Centric Infrastructure (ACI) fabric. Traffic emanating from the north or core towards the south or ACI fabric will reach the default or shared VRF. The routes are identified for the traffic to reach the host located in the ACI fabric by leaking of host routes from tenant VRFs into the default VRF or internet VRF. This functionality is supported only on Inspur CN12700 F3-Series I/O modules.

The show ip route command outputs for a scenario in which the routes from VRF A are leaked to VRF B are as given below. The output logs display the leaked routes along with the source VRF from which the routes were leaked.

```
device# show ip route 100.1.1.0/24
vrf VRF-A IP Route Table for VRF
"VRF-A"
'*' denotes best ucast
next-hop '*' denotes best
mcast next-hop '[x/y]'
denotes [preference/metric]
'%<string>' in via output denotes VRF
<string> 100.1.1.0/24, ubest/mbest:
1/0 time
*via 10.1.0.34%default, [20/0], 19:34:43, bgp-200, external, tag
6500 (evpn), segid: 2850822 tunnelid: 0xa010022 encap: VXLAN
```

```
device# show ip route 100.1.1.0/24
vrf VRF-B IP Route Table for VRF
"VRF-B"
'*' denotes best ucast
next-hop '*' denotes best
mcast next-hop '[x/y]'
denotes [preference/metric]
'%<string>' in via output denotes VRF
<string> 100.1.1.0/24, ubest/mbest:
1/0 time
```

```
*via 10.1.0.34%default, [20/0], 19:34:50, bgp-200, external, tag
6500 (evpn), segid: 2850822 tunnelid: 0xa010022 src vrf: VRF-A
encap: VXLAN
```

- Configuring Route Leaking.
- Configuring Routes Imported from a VPN to Leak into the Default VRF.
- Configuring Routes Leaked from the Default-VRF to Export to a VPN.

- Configuring Routes Imported from a VPN to Export to a VRF.
- Configuring Routes Imported from a VRF to Export to a VPN.
- Configuration Examples.
- Displaying Centralized Route Leaking Information.
- Displaying Centralized Route Leaking Information.
- Additional References.
- Feature History for Centralized VRF Route Leaking.

11.1 Configuring Route Leaking

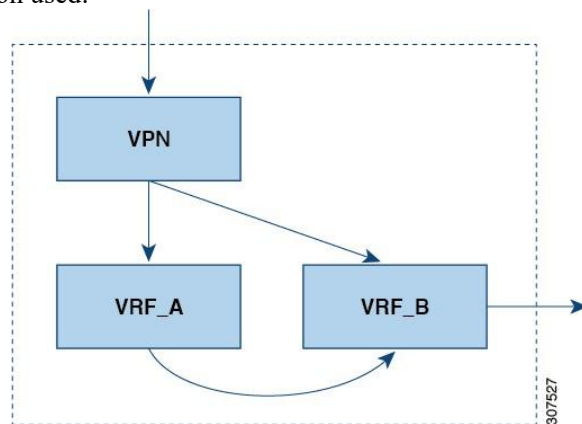
11.1.1 Licensing Requirements for Centralized VRF Route-Leaking

BGP and VXLAN require a LAN_ENTERPRISE_SERVICES_PKG license. MPLS requires an MPLS_PKG license.

11.1.2 Guidelines and Limitations for Centralized VRF Route-Leaking

The following are the guidelines and limitations for Centralized VRF Route-Leaking:

- Each prefix must be imported into each VRF for full cross-VRF reachability.
- The **feature bgp** command is required for the **export vrf default** command.
- If a VTEP has a less specific local prefix in its VRF, the VTEP might not be able to reach a more specific prefix in a different VRF.
 - VXLAN routing in hardware and packet reencapsulation at VTEP is required for Centralized VRF Route-Leaking with BGP EVPN.
 - The **feature bgp** command has to be used to enable usage of the **export vrf** and **import vrf** commands.
 - Duplicate paths pointing to the same remote path may exist in case there are VRFs on which reimporting of routes has been enabled along with the same route-targets. This may have an impact on the performance and memory.
 - There may be path duplication due to improper usage of route targets. Consider the following topology followed by the configuration used.



```
vrf context VRF_A
 address-family ipv4
  unicast route-
  target both RT_A
  export vrf allow-
  vpn
vrf context VRF_B
 address-family ipv4
```

```
unicast route-
target import RT_A
```

This configuration leads to VRF_B importing the same route via the VPN and VRF_A. This results in loss of multi-path as there are two separate paths in VRF_B having the same next hop but with different source route distinguishers.

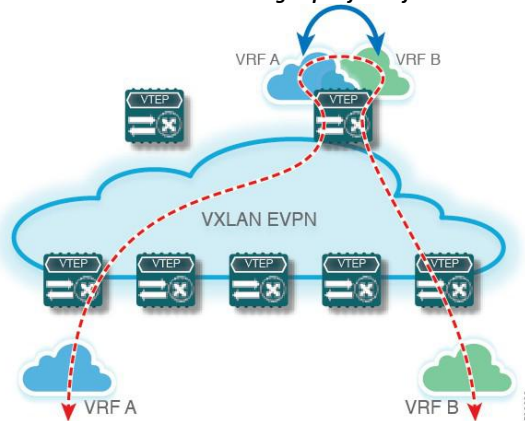
- Be careful when using local route leaking in a leaf-to-leaf case, where border-leaf routers (BLs) are leaking between the same VRFs. This scenario is more prone to routing loops. We recommend using inbound route-maps to exclude the imported routes from other BLs.
- After a remote path gets withdrawn, it can take up to 20 seconds more for BGP to completely clean up the path.

11.1.3 Centralized VRF Route-Leaking Brief - Specific Prefixes Between Custom VRF

Some pointers are given below:

- The Centralized VRF route-leaking for VXLAN BGP EVPN fabrics is depicted within Figure 2.
- BGP EVPN prefixes are cross-VRF leaked by exporting them from VRF Blue with an import into VRF Red and vice-versa. The Centralized VRF route-leaking is performed on the centralized Routing-Block (RBL) and could be any or multiple VTEPs.
- Configured less specific prefixes (aggregates) are advertised from the Routing-Block to the remaining VTEPs in the respective destination VRF.
- BGP EVPN does not export prefixes that were previously imported to prevent the occurrence of routing loops.

Figure 28 : Centralized VRF Route-Leaking - Specific Prefixes with Custom VRF



11.1.4 Configuring Centralized VRF Route-Leaking-Specific Prefixes between Custom VRF

Configuring VRF Context on the Routing-Block VTEP

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enter global configuration mode.
Step 2	vrf context vrf-name	Configure the VRF.

Step 3	vni number	Specify the VNI. The VNI associated with the VRF is often referred to as Layer-3 VNI, L3VNI or L3VPN. The L3VNI is configured as common identifier across the participating VTEPs.
Step 4	rd auto	Specify the VRFs Route Distinguisher (RD). The RD uniquely identifies a VTEP within a L3VNI.
Step 5	address-family ipv4 unicast	Configure the IPv4 Unicast address-family. Required for IPv4 over VXLAN with IPv4 underlay.
Step 6	route-target both {auto as:vni}	Configure the Route Target (RT) for import/export of IPv4 prefixes within the IPv4 unicast address-family. The Route Target (RT) is used for a per-VRF prefix import/export policy. If <i>as:vni</i> is entered, the value is in the format of ASN:NN, ASN4:NN, or IPv4:NN.
Step 7	route-target both {auto as:vni } evpn	Configure the Route Target (RT) for import/export of IPv4 prefixes within the IPv4 unicast address-family. The Route Target (RT) is used for a per-VRF prefix import/export policy. If <i>as:vni</i> is entered, the value is in the format of ASN:NN, ASN4:NN, or IPv4:NN.
Step 8	route-target import rt-from-different-vrf	Configure the Route Target (RT) for importing IPv4 prefixes from the leaked-from VRF (ie AS:VNI).
Step 9	route-targetimport rt-from-different-vrfevpn	Configure the Route Target (RT) for importing IPv4 prefixes from the leaked-from VRF (ie AS:VNI).

Configuring the BGP VRF instance on the Routing-Block

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	router bgp autonomous-system number	Configure BGP.
Step 3	vrf vrf-name	Specify the VRF.
Step 4	address-family ipv4 unicast	Configure address family for IPv4

Step 5	advertise l2vpn evpn	Enable the advertisement of EVPN routes within IPv4 address-family.
Step 6	aggregate-address <i>prefix/mask</i>	Create less specific prefix aggregate into the destination VRF.
Step 7	maximum-paths <i>ibgp number</i>	Enabling equal cost multipathing (ECMP) for iBGP prefixes.
Step 8	maximum-paths <i>number</i>	Enabling equal cost multipathing (ECMP) for eBGP prefixes

Example - Configuration Centralized VRF Route-Leaking - Specific Prefixes Between Custom VRF

Configuring VXLAN BGP EVPN Routing-Block

The VXLAN BGP EVPN Routing-Block acts as centralized route-leaking point. The leaking configuration is localized such that control-plane leaking and data-path forwarding follow the same path. Most significantly is the VRF configuration of the Routing-Block and the advertisement of the less specific prefixes (aggregates) into the respective destination VRFs.

```

vrf context Blue
  vni 51010
  rd auto
  address-family
    ipv4 unicast
    route-target
      both auto
    route-target
      both auto evpn
    route-target import
      65002:51020 route-target
    import 65002:51020 evpn
  !
vlan 2110
  vn-segment 51010
  !
interface Vlan2110 no shutdown
  mtu 9216
  vrf member Blue no ip redirects ip forward
  !
vrf context Red vni 51020
  rd auto
  address-family
    ipv4 unicast
    route-target
      both auto
    route-target
      both auto evpn
    route-target import
      65002:51010 route-target
    import 65002:51010 evpn
  !
vlan 2120
  vn-segment 51020
  !
interface Vlan2120 no shutdown
  mtu 9216
  vrf member Blue no ip redirects ip forward
  !

```

```

interface nve1 no shutdown
  host-reachability
  protocol      bgp
  source-interface
  loopback1    member
  vni          51010
  associate-vrf
  member vni 51020
  associate-vrf
!
router bgp 65002 vrf Blue

  address-family    ipv4
  unicast            advertise
  l2vpn evpn aggregate-
  address 10.20.0.0/16
  maximum-paths ibgp 2
  Maximum
  -paths 2
vrf Red
  address-family    ipv4
  unicast            advertise
  l2vpn evpn aggregate-
  address 10.10.0.0/16
  maximum-paths ibgp 2
  Maximum-paths 2

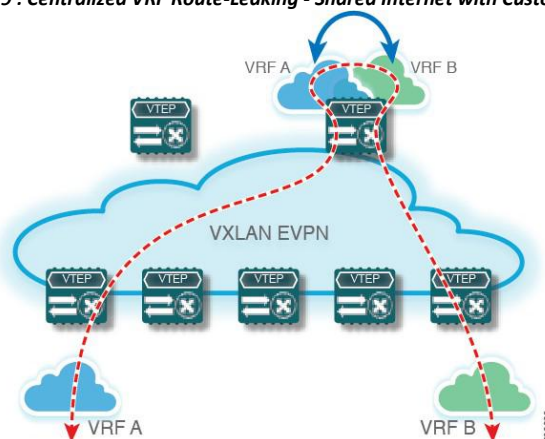
```

Centralized VRF Route-Leaking Brief - Shared Internet with Custom VRF

Some pointers follow:

- The Shared Internet with VRF route-leaking for VXLAN BGP EVPN fabrics is depicted in the following figure.
- The default-route is made exported from the Shared Internet VRF and re-advertisement within VRF Blue and VRF Red on the Border Node.
- Ensure the default-route in VRF Blue and VRF Red is not leaked to the Shared Internet VRF.
- The less specific prefixes for VRF Blue and VRF Red are exported for the Shared Internet VRF and re-advertised as necessary.
- Configured less specific prefixes (aggregates) that are advertised from the Border Node to the remaining VTEPs to the destination VRF (Blue or Red).
- BGP EVPN does not export prefixes that were previously imported to prevent the occurrence of routing loops.

Figure 29 : Centralized VRF Route-Leaking - Shared Internet with Custom VRF



11.1.5 Configuring Centralized VRF Route-Leaking-Shared Internet with Custom VRF

Configuring Internet VRF on Border Node

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enter global configuration mode.
Step 2	vrf context <i>vrf-name</i>	Configure the VRF.
Step 3	vni <i>number</i>	Specify the VNI. The VNI associated with the VRF is often referred to as Layer-3 VNI, L3VNI or L3VPN. The L3VNI is configured as common identifier across the participating VTEPs.
Step 4	ip route 0.0.0.0/0 <i>next-hop</i>	Configure default-route in shared internet VRF to external router (example).
Step 5	rd auto	Specify the VRFs Route Distinguisher (RD). The RD uniquely identifies a VTEP within a L3VNI.
Step 6	address-family ipv4 unicast	Configure the IPv4 Unicast address-family. Required for IPv4 over VXLAN with IPv4 underlay.
Step 7	route-target both { auto <i>as:vni</i> }	Configure the Route Target (RT) for import/export of EVPN and IPv4 prefixes within the IPv4 unicast address-family.
Step 8	route-target both <i>shared-vrf-rt evpn</i>	Configure a special Route Target (RT) for the import/export of the shared IPv4 prefixes. Additional import/export map for further qualification is supported

Configuring Shared Internet BGP Instance on the Border Node

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	router bgp <i>autonomous-system number</i>	Configure BGP.
Step 3	vrf <i>vrf-name</i>	Specify the VRF.

Step 4	address-family ipv4 unicast	Configure address family for IPv4
Step 5	advertise l2vpn evpn	Enable the advertisement of EVPN routes within IPv4 address-family.
Step 6	aggregate-address <i>prefix/mask</i>	Create less specific prefix aggregate into the destination VRF.
Step 7	maximum-paths ibgp <i>number</i>	Enabling equal cost multipathing (ECMP) for iBGP prefixes.
Step 8	maximum-paths <i>number</i>	Enabling equal cost multipathing (ECMP) for eBGP prefixes.

Configuring Custom VRF on Border Node

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	ip prefix-list <i>name</i> seq 5 permit 0.0.0.0/0	Configure IPv4 prefix-list for default-route filtering.
Step 3	route-map <i>name</i> deny 10	Create route-map with leading deny statement to prevent the default-route of being leaked.
Step 4	match ip address prefix-list <i>name</i>	Match against the IPv4 prefix-list that contains the default-route.
Step 5	route-map <i>name</i> permit 20	Create route-map with trailing allow statement to advertise non-matching routes via route-leaking.

Configuring Custom VRF Context on the Border Node - 1

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	vrf context <i>vrf-name</i>	Configure the VRF.
Step 3	vni <i>number</i>	Specify the VNI. The VNI associated with the VRF is often referred to as Layer-3 VNI, L3VNI or L3VPN. The L3VNI is configured as the common identifier across the participating VTEPs.

Step 4	rd auto	Specify the VRFs Route Distinguisher (RD). The Route Distinguisher (RD) uniquely identifies a VTEP within a L3VNI.
Step 5	ip route 0.0.0.0/0 Null0	Configure default-route in common VRF to attract traffic towards Border Node with Shared Internet VRF.
Step 6	address-family ipv4 unicast	Configure the IPv4 Unicast address-family. Required for IPv4 over VXLAN with IPv4 underlay.
Step 7	route-target both {auto as:vni}	Configure the Route Target (RT) for import/export of IPv4 prefixes within the IPv4 unicast address-family The Route Target (RT) is used for a per-VRF prefix import/export policy. If as:vni is entered, the value is in the format of ASN:NN, ASN4:NN, or IPv4:NN.
Step 8	route-target both {auto as:vni} evpn	Configure the Route Target (RT) for import/export of IPv4 prefixes within the IPv4 unicast address-family The Route Target (RT) is used for a per-VRF prefix import/export policy. If as:vni is entered, the value is in the format of ASN:NN, ASN4:NN, or IPv4:NN.
Step 9	import map name	Apply a route-map on routes being imported into this routing table.

Configuring Custom VRF Instance in BGP on the Border Node

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	router bgp autonomous-system-number	Configure BGP.
Step 3	vrf vrf-name	Specify the VRF.
Step 4	address-family ipv4 unicast	Configure address family for IPv4.
Step 5	advertise l2vpn evpn	Enable the advertisement of EVPN routes within IPv4 address-family.
Step 6	network 0.0.0.0/0	Creating IPv4 default-route network statement.
Step 7	maximum-paths ibgp number	Enabling equal cost multipathing (ECMP) for iBGP prefixes.

Step 8	maximum-paths <i>number</i>	Enabling equal cost multipathing (ECMP) for eBGP prefixes.
---------------	------------------------------------	--

Example - Configuration Centralized VRF Route-Leaking - Shared Internet with Custom VRF

An example of Centralized VRF route-leaking with Shared Internet VRF

Configuring VXLAN BGP EVPN Border Node for Shared Internet VRF

The VXLAN BGP EVPN Border Node provides a centralized Shared Internet VRF. The leaking configuration is localized such that control-plane leaking and data-path forwarding following the same path. Most significantly is the VRF configuration of the Border Node and the advertisement of the default-route and less specific prefixes (aggregates) into the respective destination VRFs.

```

vrf context Shared
  vni 51099
  ip route 0.0.0.0/0
    10.9.9.1 rd auto
  address-family
    ipv4 unicast
    route-target
      both auto route-
target both auto
    evpn route-
target both
    99:99 route-
target both
    99:99 evpn
  !
vlan 2199
  vn-segment 51099
  !
interface Vlan2199 no shutdown
  mtu 9216
  vrf member Shared no ip redirects ip forward
  !
ip prefix-list PL_DENY_EXPORT seq 5 permit
0.0.0.0/0
  !
route-map RM_DENY_IMPORT deny 10
  match ip address prefix-list
PL_DENY_EXPORT route-map
RM_DENY_IMPORT permit 20
  !
vrf context Blue vni 51010
  ip route
0.0.0.0/0
  Null0 rd auto
  address-family
    ipv4 unicast
    route-target
      both auto route-
target both auto
    evpn route-
target both
    99:99 route-
target both
    99:99 evpn
    import map
    RM_DENY_IMPORT
  !
vlan 2110

```

```

    vn-segment 51010
    !
interface Vlan2110 no shutdown
mtu 9216
vrf member Blue no ip redirects ip forward
!
vrf context Red vni 51020
ip route
0.0.0.0/0
Null0 rd auto
address-family
  ipv4 unicast
  route-target
  both auto route-
  target both auto
  evpn route-
  target both
  99:99 route-
  target both
  99:99 evpn
  import map
  RM_DENY_IMPORT
!
vlan 2120
  vn-segment 51020
!
interface Vlan2120 no shutdown
mtu 9216

vrf member Blue no ip
redirects ip forward
!
interface nve1 no
shutdown
host-reachability
protocol bgp source-
interface loopback1
member vni 51099
associate-vrf member
vni 51010 associate-
vrf member vni 51020
associate-vrf
!
router bgp 65002 vrf
Shared
address-family ipv4
unicast advertise
l2vpn evpn aggregate-
address 10.10.0.0/16
aggregate-address
10.20.0.0/16 maximum-
paths ibgp 2
maximum
-paths 2
vrf Blue
address-family ipv4
unicast advertise
l2vpn evpn
network 0.0.0.0/0
maximum-paths
ibgp 2
maximum
-paths 2
vrf Red

```

```

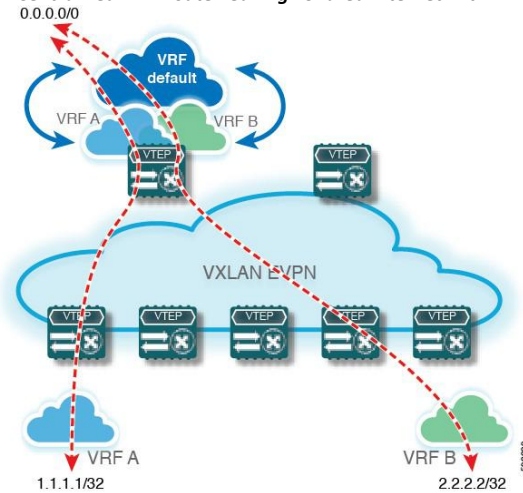
address-family ipv4
 unicast advertise
  l2vpn evpn
  network 0.0.0.0/0
  maximum-paths
  ibgp 2
  maximum-paths 2
    
```

Centralized VRF Route-Leaking Brief - Shared Internet with VRF Default

Some pointers are given below:

- The Shared Internet with VRF route-leaking for VXLAN BGP EVPN fabrics is depicted within Figure 4.
- The default-route is made exported from VRF default and re-advertisement within VRF Blue and VRF Red on the Border Node.
- Ensure the default-route in VRF Blue and VRF Red is not leaked to the Shared Internet VRF
- The less specific prefixes for VRF Blue and VRF Red are exported to VRF default and re-advertised as necessary.
- Configured less specific prefixes (aggregates) that are advertised from the Border Node to the remaining VTEPs to the destination VRF (Blue or Red).
- BGP EVPN does not export prefixes that were previously imported to prevent the occurrence of routing loops.

Figure 30 : Centralized VRF Route-Leaking - Shared Internet with VRF Default



11.1.6 Configuring Centralized VRF Route-Leaking - Shared Internet with VRF Default

Configuring VRF Default on Border Node

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	ip route 0.0.0.0/0 next-hop	Configure default-route in VRF default to external router (example)

Configuring BGP Instance for VRF Default on the Border Node

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	router bgp <i>autonomous-system number</i>	Configure BGP.
Step 3	address-family ipv4 unicast	Configure address family for IPv4.
Step 4	aggregate-address <i>prefix/mask</i>	Create less specific prefix aggregate in VRF default.
Step 5	maximum-paths <i>number</i>	Enabling equal cost multipathing (ECMP) for eBGP prefixes.

Configuring Custom VRF on Border Node

This procedure applies equally to IPv6

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	ip prefix-list <i>name seq 5 permit 0.0.0.0/0</i>	Configure IPv4 prefix-list for default-route filtering.
Step 3	route-map <i>name deny 10</i>	Create route-map with leading deny statement to prevent the default-route of being leaked.
Step 4	match ip address prefix-list <i>name</i>	Match against the IPv4 prefix-list that contains the default-route.
Step 5	route-map <i>name permit 20</i>	Create route-map with trailing allow statement to advertise non-matching routes via route-leaking.

Configuring Filter for Permitted Prefixes from VRF Default on the Border Node

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	route-map <i>name permit 10</i>	Create route-map with allow statement to advertise routes via route-leaking to the customer VRF and subsequently remote VTEPs.

Configuring Custom VRF Context on the Border Node - 2

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	vrf context <i>vrf-name</i>	Configure the VRF.
Step 3	vni <i>number</i>	Specify the VNI. The VNI associated with the VRF is often referred to as Layer-3 VNI, L3VNI or L3VPN. The L3VNI is configured as common identifier across the participating VTEPs.
Step 4	rd auto	Specify the VRFs Route Distinguisher (RD). The Route Distinguisher (RD) uniquely identifies a VTEP within a L3VNI.
Step 5	ip route 0.0.0.0/0 Null0	Configure default-route in common VRF to attract traffic towards Border Node with Shared Internet VRF.
Step 6	address-family ipv4 unicast	Configure the IPv4 Unicast address-family. Required for IPv4 over VXLAN with IPv4 underlay.
Step 7	route-target both auto <i>AS:VNI</i>	Configure the Route Target (RT) for import/export of EVPN and IPv4 prefixes within the IPv4 unicast address-family.
Step 8	route-target both auto <i>AS:VNI evpn</i>	Configure the Route Target (RT) for import/export of EVPN and IPv4 prefixes within the IPv4 unicast address-family.
Step 9	route-target both <i>shared-vrf-rt</i>	Configure a special Route Target (RT) for the import/export of the Shared IPv4 prefixes. Additional import/export map for further qualification is supported
Step 10	route-target both <i>shared-vrf-rt evpn</i>	Configure a special Route Target (RT) for the import/export of the Shared IPv4 prefixes. Additional import/export map for further qualification is supported
Step 11	import vrf default map <i>name</i>	Permits all routes, from VRF default, from being imported into the custom VRF according to the specific route-map.

Configuring Custom VRF Instance in BGP on the Border Node

This procedure applies equally to IPv6.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	router bgp <i>autonomous-system-number</i>	Configure BGP.
Step 3	vrf <i>vrf-name</i>	Specify the VRF.
Step 4	address-family ipv4 unicast	Configure address family for IPv4.
Step 5	advertise l2vpn evpn	Enable the advertisement of EVPN routes within IPv4 address-family.
Step 6	network 0.0.0.0/0	Creating IPv4 default-route network statement.
Step 7	maximum-paths ibgp <i>number</i>	Enabling equal cost multipathing (ECMP) for iBGP prefixes.
Step 8	maximum-paths <i>number</i>	Enabling equal cost multipathing (ECMP) for eBGP prefixes.

Example - Configuring Centralized VRF Route-Leaking - VRF Default with Custom VRF

An example of Centralized VRF route-leaking with VRF default

Configuring VXLAN BGP EVPN Border Node for VRF Default

The VXLAN BGP EVPN Border Node provides centralized access to VRF default. The leaking configuration is localized such that control-plane leaking and data-path forwarding following the same path. Most significantly is the VRF configuration of the Border Node and the advertisement of the default-route and less specific prefixes (aggregates) into the respective destination VRFs.

```

ip route 0.0.0.0/0 10.9.9.1
!
ip prefix-list PL_DENY_EXPORT seq 5 permit 0.0.0.0/0
!
route-map permit 10
match ip address prefix-list
PL_DENY_EXPORT route-map
RM_DENY_EXPORT permit 20
route-map RM_PERMIT_IMPORT permit 10
!
vrf context Blue vni 51010
ip route
0.0.0.0/0 Null0
rd auto
address-family ipv4
unicast route-
target both auto
route-target both
auto evpn
import vrf default map RM_PERMIT_IMPORT
export vrf default 100 map RM_DENY_EXPORT allow-vpn
!
vlan 2110

```

```

    vn-segment 51010
    !
    interface Vlan2110 no shutdown
    mtu 9216
    vrf member Blue no ip redirects ip forward
    !
    vrf context Red vni 51020
    ip route
    0.0.0.0/0
    Null0 rd auto
    address-family
    ipv4 unicast
    route-target
    both auto
    route-target
    both auto evpn
    import vrf default map RM_PERMIT_IMPORT
    export vrf default 100 map RM_DENY_EXPORT
    allow-vpn
    !
    vlan 2120
    vn-segment 51020
    !
    interface Vlan2120 no shutdown
    mtu 9216
    vrf member Blue no ip redirects ip forward
    !
    interface nve1 no shutdown
    host-reachability
    protocol bgp
    source-interface
    loopback1 member
    vni 51010
    associate-vrf
    member vni 51020
    associate-vrf
    !
    router bgp 65002
    address-family ipv4
    unicast aggregate-
    address
    10.10.0.0/16
    aggregate-address
    10.20.0.0/16
    maximum-paths 2
    maximum-
    paths ibgp 2
    vrf Blue
    address-family
    ipv4 unicast
    advertise
    l2vpn evpn
    network
    0.0.0.0/0
    maximum-paths
    ibgp 2
    maxi
    mum-
    paths 2
    vrf Red
    address-family
    ipv4 unicast
    advertise

```

```

l2vpn      evpn
network
0.0.0.0/0
maximum-paths
ibgp 2
maximum-paths 2

```

11.2 Configuring Routes Imported from a VPN to Leak into the Default VRF

You can configure a VRF to allow routes that are imported from a BGP VPN to be exported to the default VRF. Use this procedure for a non-default VRF. Use this procedure for a non-default VRF.

If you have not already enabled BGP, enable it now using the **feature bgp** command.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	vrf context <i>vrf-name</i>	Creates a new VRF and enters VRF configuration mode. The name can be any case-sensitive, alphanumeric string up to 32 characters.
Step 3	address-family ipv4 unicast	Configure the address family for IPv4.
Step 4	export vrf default [<i>prefix-limit</i>] map <i>route-map</i> allow-vpn	Configures the current VRF to allow routes that are imported from a BGP VPN to be exported to the default VRF.

11.3 Configuring Routes Leaked from the Default-VRF to Export to a VPN

You can configure a VRF to allow routes leaked from the default VRF to be exported to a BGP VPN. Use this procedure for a non-default VRF.

If you have not already enabled BGP, enable it now using the **feature bgp** command.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	vrf context <i>vrf-name</i>	Creates a new VRF and enters VRF configuration mode. The name can be any case-sensitive, alphanumeric string up to 32 characters.
Step 3	address-family <i>address-family sub-family</i>	Configure the address family for IPv4.
Step 4	import vrf default [<i>prefix-limit</i>] map <i>route-map</i> advertise-vpn	Configures the current VRF to allow routes imported from the default VRF to be exported to a BGP VPN.

11.4 Configuring Routes Imported from a VPN to Export to a VRF

You can configure a VRF to allow VPN imported routes to be exported to another VRF. Use this procedure for non-default VRFs.

If you have not already enabled BGP, enable it now using the **feature bgp** command.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	vrf context <i>vrf-name</i>	Creates a new VRF and enters VRF configuration mode. The name can be any case-sensitive, alphanumeric string up to 32 characters.
Step 3	address-family <i>address-family sub-family</i>	Configure the address family for IPv4.
Step 4	export vrf allow-vpn	Configures a VRF to allow routes imported from a BGP VPN to be exported to a non-default VRF.

11.5 Configuring Routes Imported from a VRF to Export to a VPN

You can configure a VRF to allow routes imported from another VRF to be exported to a BGP VPN. Use this procedure for non-default VRFs.

If you have not already enabled BGP, enable it now using the **feature bgp** command.

Procedure

	Command or Action	Purpose
Step 1	configure terminal	Enters global configuration mode.
Step 2	vrf context <i>vrf-name</i>	Creates a new VRF and enters VRF configuration mode. The name can be any case-sensitive, alphanumeric string up to 32 characters.
Step 3	address-family <i>address-family sub-family</i>	Configure the address family for IPv4.
Step 4	import vrf advertise-vpn	Configures the current VRF to allow routes that are imported from another VRF to be exported to a BGP VPN.

11.6 Configuration Examples

Configuring BGP VPN to Default VRF Reachability

In this example, the configuration enables route re-importation through an intermediate VRF, called VRF_A, which is between the VPN and the default VRF.

```
vrf context VRF_A
  address-family ipv4
    unicast route-
      target both auto
  evpn
  import vrf default map MAP_1
  advertise-vpn export vrf default
  map MAP_1 allow-vpn
```

Route re-importation is enabled by using the **advertise-vpn** option to control importing routes from the VPN into VRF_A, and **allow-vpn** for the export map to control exporting VPN-imported routes from VRF_A out to the default VRF. Configuration occurs on the intermediate VRF.

Configuring VPN to VRF-Lite Reachability

In this example, the VPN connects to a tenant VRF, called VRF_A. VRF_A connects a VRF-Lite, called VRF-B. The configuration enables VPN imported routes to be leaked from VRF_A to VRF_B.

```
vrf context VRF_A
  address-family ipv4
    unicast route-
      target both auto
    route-target both
  auto evpn route-
    target import 3:3
  route-target
  export 2:2
  import vrf
  advertise-vpn
  export vrf
  allow-vpn
vrf context VRF_B
  address-family ipv4
    unicast route-
      target both 1:1
  route-target import 2:2
  route-target export 3:3
```

Route leaking between the two is enabled by using the **allow-vpn** in an export map configured in VRF_A (tenant). The export map in VRF_A allows route imported from the VPN to be leaked into the VRF_B. Routes processed by the export map have the **route-map export** and **export-map** attributes added to the route's set of route targets. The import map uses **advertise-vpn** which enables routes that are imported from the VRF-Lite for be exported out to the VPN.

After a route leaks between the VRFs, it is reoriginated and its route targets are replaced by the route target export and export map attributes specified by the new VRF's configuration.

Leaf-to-Leaf Reachability

In this example, two VPNs exist and two VRFs exist. VPN_1 is connected to VRF_A and VPN_2 is connected to VRF_B. Both VRFs are route distinguishers (RDs).

```
vrf context VRF_A
  address-family ipv4
    unicast route-
      target both auto
    route-target both
  auto evpn route-
```

```

target import 3:3
route-target
export 2:2
import vrf
advertise-vpn
export vrf
allow-vpn
vrf context VRF_B
address-family ipv4
unicast route-
target both 1:1
route-target import 2:2
route-target
export 3:3
import vrf
advertise-vpn
export vrf
allow-vpn

```

Route leaking between the two is enabled by using the **allow-vpnin** an export map configured in VRF_A and VRF_B. VPN imported routes have **route-map** export and **export-map** attributes added to the route's set of route targets. The import map uses **advertise-vpn** option which enables routes that are imported from each VRF to be exported out to the VPN.

After a route leaks between the VRFs, it is reoriginated and its route targets are replaced by the route target export and export map attributes specified by the new VRF's configuration.

Leaf-to-Leaf with Loop Prevention

In the leaf-to-leaf configuration, you can inadvertently cause loops between the BLs that are leaking between the same VRFs unless you are careful with your route maps:

- You can use an inbound route map in each BL to deny updates from every other BL.
- If a BL originates a route, a standard community can be applied, which enables other BLs to accept the routes.

This community is then stripped in the receiving BL.

In the following example, VTEPs 3.3.3.3, 4.4.4.4 and 5.5.5.5 are the BLs.

```

ip prefix-list BL_PREFIX_LIST seq 5 permit
3.3.3.3/32 ip prefix-list BL_PREFIX_LIST
seq 10 permit 4.4.4.4/32 ip prefix-list
BL_PREFIX_LIST seq 20 permit 5.5.5.5/32
ip community-list standard BL_COMMUNITY seq 10
permit 123:123 route-map INBOUND_MAP permit 5
match community
BL_COMMUNITY set
community none
route-map INBOUND_MAP deny 10
match ip next-hop prefix-list
BL_PREFIX_LIST route-map
INBOUND_MAP permit 20
route-map
OUTBOUND_SET_COMM permit
10 match evpn route-type 2
mac-ip
set community 123:123
route-map
SET_COMM permit
10 set community
123:123
route-map allow permit 10

```



```

vrf context vni100 vni 100
address-family ipv4 unicast route-target
import 2:2
route-target export 1:1 route-target both
auto route-target both auto evpn import vrf
advertise-vpn export vrf allow-vpn

vrf context vni200 vni 200
address-family ipv4 unicast route-target
import 1:1
route-target export 2:2 route-target both
auto route-target both auto evpn import vrf
advertise-vpn export vrf allow-vpn

router bgp 100
template peer
rr remote-as
100
update-source loopback0

address-family l2vpn evpn send-community
send-community extended route-map
INBOUND_MAP in
route-map OUTBOUND_SET_COMM out neighbor
101.101.101.101 inherit peer rr
neighbor 102.102.102.102 inherit peer rr
vrf vni100
address-family ipv4 unicast
network 3.3.3.100/32 route-map
SET_COMM vrf vni200
address-family ipv4 unicast
network 3.3.3.200/32 route-map SET_COMM

```

In this example, the tenant VRFs for the border leaf (BL) router can leak traffic by enabling extra import export flows, and the route targets in the route maps determine where the routes are imported from or exported to.

Multipath in a VRF

In this example, a VPN has multiple incoming paths. This configuration enables route leaking through an intermediate VRF, called VRF_A, which is between the VPN and another VRF, named VRF_B. Assume that multipathing is enabled in VRF_A.

```

vrf context VRF_A
address-family ipv4
unicast route-
target both auto
evpn route-target
export 3:3 export
vrf allow-vpn
vrf context VRF_B
address-family ipv4
unicast route-
target import 3:3

```

Route leaking is enabled by **allow-vpn** in the export map configured in VRF_A. When two paths for a given prefix are learnt from a VPN and imported into VRF_A, two different paths exist in VRF_B with the same source RD (VRF_A's local RD). Each route is distinguished by the original source RD (remote RD).

Path Duplication

In this example, the configuration enables a single VPN path to be imported into both VRF_A and VRF_B. Because VRF_A is configured with export **vrf allow-vpn**, VRF_A also leaks its routes into VRF_B. VRF_B then has two paths with same source RD (VRF_A's local RD), each one distinguished by the original source RD (remote RD).

```

vrf context VRF_A
 address-family ipv4
  unicast route-target
  import 1:1 evpn
  route-target export
  1:1 evpn route-
  target export 2:2
 export vrf
 allow-vpn
vrf context
VRF_B
 address-family ipv4
  unicast route-target
  import 1:1 evpn
  route-target import
  2:2

```

This configuration creates a situation in which multipathing does not exist.

11.7 Displaying Centralized Route Leaking Information

The following table shows the commands that display information about the Centralized Route Leaking feature.

Command	Action
show bgp vrf <i>vrf-name</i> process	For a default or non-default VRF, shows the enabled state (Yes or No) of the import advertise-vpn and export allow-vpn options.
show bgp vrf <i>vrf-name</i> ipv4 unicast <i>prefix</i>	Shows information about imported paths, including a list of destinations a route has been imported from.

Route re-importation is enabled by using the **advertise-vpn** option to control importing routes from the VPN into VRF_A, and **allow-vpn** for the export map to control exporting VPN-imported routes from VRF_A out to the default VRF. Configuration occurs on the intermediate VRF.

Configuring VPN to VRF-Lite Reachability

In this example, the VPN connects to a tenant VRF, called VRF_A. VRF_A connects a VRF-Lite, called VRF-B. The configuration enables VPN imported routes to be leaked from VRF_A to VRF_B.

```

vrf context VRF_A
 address-family ipv4 unicast route-target both
 auto route-target both auto evpn route-target
 import 3:3
 route-target export 2:2 import vrf advertise-
 vpn export vrf allow-vpn
vrf context VRF_B
 address-family ipv4 unicast route-target both
 1:1
 route-target import 2:2
 route-target export 3:3

```

Route leaking between the two is enabled by using the **allow-vpn** in an export map configured in VRF_A (tenant). The export map in VRF_A allows route imported from the VPN to be leaked into the VRF_B. Routes processed by the export map have the **route-map** export and **export-map** attributes added to the route's set of route targets. The import map uses **advertise-vpn** which enables routes that are imported from the VRF-Lite for be exported out to the VPN.

After a route leaks between the VRFs, it is reoriginated and its route targets are replaced by the route target

export and export map attributes specified by the new VRF's configuration.

Leaf-to-Leaf Reachability

In this example, two VPNs exist and two VRFs exist. VPN_1 is connected to VRF_A and VPN_2 is connected to VRF_B. Both VRFs are route distinguishers (RDs).

```
vrf context VRF_A
  address-family
    ipv4    unicast
    route-target
      both  auto
    route-target
      both auto evpn

    route-target import 3:3
    route-target
    export      2:2
    import      vrf
    advertise-vpn
    export      vrf
    allow-vpn
vrf context VRF_B
  address-family ipv4
    unicast    route-
    target both 1:1
    route-target import 2:2
    route-target
    export      3:3
    import      vrf
    advertise-vpn
    export      vrf
    allow-vpn
```

Route leaking between the two is enabled by using the **allow-vpn** in an export map configured in VRF_A and VRF_B. VPN imported routes have **route-map export** and **export-map** attributes added to the route's set of route targets. The import map uses **advertise-vpn** option which enables routes that are imported from each VRF to be exported out to the VPN.

After a route leaks between the VRFs, it is reoriginated and its route targets are replaced by the route target export and export map attributes specified by the new VRF's configuration.

Leaf-to-Leaf with Loop Prevention

In the leaf-to-leaf configuration, you can inadvertently cause loops between the BLs that are leaking between the same VRFs unless you are careful with your route maps:

- You can use an inbound route map in each BL to deny updates from every other BL.
- If a BL originates a route, a standard community can be applied, which enables other BLs to accept the routes.

This community is then stripped in the receiving BL.

In the following example, VTEPs 3.3.3.3, 4.4.4.4 and 5.5.5.5 are the BLs.

```
ip prefix-list BL_PREFIX_LIST seq 5 permit
3.3.3.3/32 ip prefix-list BL_PREFIX_LIST seq
10 permit 4.4.4.4/32 ip prefix-list
BL_PREFIX_LIST seq 20 permit 5.5.5.5/32
ip community-list standard BL_COMMUNITY seq 10 permit
123:123 route-map INBOUND_MAP permit 5
match community
BL_COMMUNITY set
community none
```

```

route-map INBOUND_MAP deny 10
match ip next-hop prefix-list
BL_PREFIX_LIST route-map INBOUND_MAP
permit 20
route-map OUTBOUND_SET_COMM
permit 10 match evpn route-
type 2 mac-ip
set community 123:123
route-map SET_COMM
permit 10 set
community 123:123
route-map allow permit 10

vrf context vni100 vni 100
address-family ipv4 unicast route-target import 2:2
route-target export 1:1 route-target both auto route-target both auto evpn import
vrf advertise-vpn export vrf allow-vpn

vrf
context
vni200 vni
200

address-family ipv4 unicast route-target import 1:1
route-target export 2:2 route-target both auto route-target both auto evpn import
vrf advertise-vpn export vrf allow-vpn

router bgp 100 template peer rr remote-as 100
update-source loopback0 address-family l2vpn evpn send-community
send-community extended route-map INBOUND_MAP in
route-map OUTBOUND_SET_COMM out neighbor 101.101.101.101 inherit peer rr
neighbor 102.102.102.102 inherit peer rr
vrf vni100
address-family ipv4 unicast
network 3.3.3.100/32 route-
map SET_COMM vrf vni200
address-family ipv4 unicast
network 3.3.3.200/32 route-map SET_COMM

```

In this example, the tenant VRFs for the border leaf (BL) router can leak traffic by enabling extra import export flows, and the route targets in the route maps determine where the routes are imported from or exported to.

Multipath in a VRF

In this example, a VPN has multiple incoming paths. This configuration enables route leaking through an intermediate VRF, called VRF_A, which is between the VPN and another VRF, named VRF_B. Assume that multipathing is enabled in VRF_A.

```

vrf context VRF_A
address-family
ipv4 unicast
route-target
both auto evpn
route-target
export 3:3
export vrf
allow-vpn
vrf context VRF_B
address-family
ipv4 unicast
route-target
import 3:3

```

Route leaking is enabled by **allow-vpn** in the export map configured in VRF_A. When two paths for a given prefix are learnt from a VPN and imported into VRF_A, two different paths exist in VRF_B with the same source RD

(VRF_A's local RD). Each route is distinguished by the original source RD (remote RD).

Path Duplication

In this example, the configuration enables a single VPN path to be imported into both VRF_A and VRF_B. Because VRF_A is configured with **export vrf allow-vpn**, VRF_A also leaks its routes into VRF_B. VRF_B then has two paths with same source RD (VRF_A's local RD), each one distinguished by the original source RD (remote RD).

```
vrf context VRF_A
  address-family ipv4 unicast
    route-target import
    1:1 evpn route-
    target export 1:1
    evpn route-target
    export 2:2
  export vrf
  allow-vpn
  vrf context
  VRF_B
    address-family ipv4
    unicast route-target
    import 1:1 evpn
    route-target import
    2:2
```

This configuration creates a situation in which multipathing does not exist.

11.8 Displaying Centralized Route Leaking Information

To display information about the RTs that have been configured and to check if the **allow-vpn** and **advertise-vpn** keywords have been configured, use the **show bgp vrfvrf process** command.

```
switch# show bgp vrf vni100
process Information regarding
configured VRFs: BGP
Information for VRF vni100
VRF Id : 3
VRF state :
UP VNID :
100 (valid)
Topo Id :
100 Encap
type :
VXLAN VTEP
IP :
3.3.3.3
VTEP Virtual IP :
0.0.0.0 VTEP VIP-
R : 0.0.0.0
Router-MAC : 5254.000e.7996
VIP Derived MAC :
5254.000e.7996 VIP-R
Derived MAC :
0000.0000.0000 Router-ID :
3.3.3.100
Configured Router-ID :
0.0.0.0 Confed-ID : 0
Cluster-ID : 0.0.0.0
No. of configured
peers : 0 No. of
pending config peers :
0 No. of established
```

```

peers : 0 VRF RD :
3.3.3.3:3
Information for address family IPv4 Unicast in
VRF vni100 Table Id : 0x3
Table state : UP
Peers Active-peers Routes Paths Networks
Aggregates 0 0 11 11 1 0
Redistribution None
Auto RT is configured EVPN Auto RT is
configured Export RT list:
1:1 100:100
Import RT list: 2:2 100:100
EVPN Export RT list: 100:100
EVPN Import RT list: 100:100
MVPN Export RT list: 100:100
MVPN Import RT list: 100:100
Label mode: per-vrf
Import default limit :
1000 Import default
prefix count : 1

Import default map : allow
Import default
advertise-vpn : Yes
Import VRF advertise-
vpn : Yes Export
default limit : 1000
Export default prefix
count : 6
Export default map :
NO_DEFAULT_ROUTE Export
default allow-vpn : Yes
Export VRF
allow-vpn : Yes
Nexthop
trigger-delay
critical 3000
ms
non-critical 10000 ms

```

To verify if a specific route is being leaked from VRF to another, use the **show bgp vrf vrf ipv4 unicast prefix** command and check the 'Imported to <number-of-destinations> destinations' field in the output logs.

```

switch# show bgp vrf vni100 ipv4 unicast
4.4.4.100
BGP routing table information for VRF vni100, address family
IPv4 Unicast BGP routing table entry for 4.4.4.100/32, version
50
Paths: (1 available, best #1)
Flags: (0x8008021e) (high32 00000000) on xmit-list, is in urib, is
best urib route, is in HW
vpn: version 97, (0x100002)
on xmit-list Multipath: iBGP
Advertised path-id 1, VPN AF advertised path-
id 1
Path type: internal, path is valid, is best path, no labeled
nexthop, in rib Imported from
4.4.4.4:3:[5]:[0]:[0]:[32]:[4.4.4.100]/224
Imported to 2
destination(s) Imported
paths list: vni200
default
AS-Path: NONE, path sourced internal to AS
4.4.4.4 (metric 81) from 101.101.101.101 (101.101.101.101)

```

```
Origin IGP, MED not set, localpref
100, weight 0 Received label 100
Extcommunity: RT:1:1 RT:100:100 ENCAP:8 Router
MAC:5254.00cd.a816 Originator: 4.4.4.4 Cluster list:
101.101.101.101
VRF advertise information:
Path-id 1 not advertised
to any peer VPN AF
advertise information:
Path-id 1 not advertised to any peer
```

11.9 Feature History for Centralized VRF Route Leaking

Table 9 : Feature History for IP TCP MSS

Feature Name	Release	Feature Information
Centralized VRF Route Leaking	8.4(1)	VXLAN BGP EVPN uses MP-BGP and its route-policy concept to import and export prefixes. The ability of this very extensive route-policy model allows to leak routes from one VRF to another VRF and vice-versa; any combination of custom VRF or VRF default can be used. VRF route-leaking is a switch-local function at specific to a location in the network, the location where the cross-VRF route-target import/export configuration takes place (leaking point). The forwarding between the different VRFs follows the control-plane, the location of where the configuration for the route-leaking is performed- hence Centralized VRF route-leaking. With the addition of VXLAN BGP EVPN, the leaking point requires to advertise the cross-VRF imported/exported route and advertise them towards the remote VTEPs or External Routers.