BGP EVPN VXLAN Configuration Guide, Cisco IOS XE Gibraltar 16.12.x (Catalyst 9600 Switches)

First Published: 2019-08-01

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

CONTENTS

CHAPTER 1

BGP EVPN VXLAN Overview

BGP EVPN VXLAN

BGP EVPN VXLAN is a campus network solution for Cisco Catalyst 9000 Series Switches running Cisco IOS XE software. This solution is a result of proposed IETF standards and Internet drafts submitted by the BGP Enabled ServiceS (bess[1]) workgroup and is designed to provide a unified overlay network solution, and also address the challenges and drawbacks of existing technologies.

This chapter provides a background for the solution's evolution and covers conceptual information and basic terminology that is required to understand BGP EVPN VXLAN. Later chapters cover configuration, implementation, functionalities, and troubleshooting information for BGP EVPN VXLAN.

The Evolution of BGP EVPN VXLAN

Traditionally, VLANs have been the standard method for providing network segmentation in campus networks. VLANs use loop prevention techniques such as Spanning Tree Protocol (STP), which impose restrictions on network design and resiliency. Further, because there is a limitation with the number of VLANs that can be used to address layer 2 segments (4094 VLANs), VLANs are a limiting factor for IT departments and cloud providers who build large and complex campus networks.

VXLAN is designed to overcome the inherent limitations of VLANs and STP. It is a proposed IETF standard [RFC 7348] to provide the same Ethernet Layer 2 network services as VLANs do, but with greater flexibility. Functionally, it is a MAC-in- UDP encapsulation protocol that runs as a virtual overlay on an existing Layer 3 network.

However, VXLAN by itself does not provide for optimal switching and routing in a network, because the "flood and learn" mechanism it uses, limits its scalability (for a host to be reachable, the host's information is flooded across the network). A VXLAN overlay, requires:

- An underlying transport network that performs data plane forwarding, for unicast communication between end points connected to the fabric.

- A control plane that is capable of distributing Layer 2 and Layer 3 host reachability information across the network.

To meet these additional requirements, Internet drafts submitted by the bess workgroup (draft-ietf-bess-evpn-overlay-12), propose MP-BGP, which features Network Layer Reachability Information (NLRI), to carry both Layer 2 MAC and Layer 3 IP information at the same time. With MAC and IP information available together for forwarding decisions, routing and switching within a network is optimised. This also minimizes the use of the conventional flood and learn mechanism, which limits the VXLAN fabric's ability to scale. The extension that allows BGP to transport Layer 2 MAC and Layer 3 IP information is EVPN.

Benefits of Deploying an Overlay-Underlay Architecture using BGP EVPN VXLAN

Deploying an overlay-underlay architecture using BGP EVPN VXLAN provides the following advantages:

- 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.
- Flexibility — VXLAN allows workloads to be placed anywhere, along with the traffic separation required in a multitenant environment. The traffic separation is done by network segmentation using VXLAN segment IDs or VXLAN Network Identifiers (VNIs). Workloads for a tenant can be distributed across different physical devices but they are identified by their respective Layer 2 VNI or Layer 3 VNI.
- Mobility — Virtual machines can be moved from one location to another without updating spine switch tables. This is because entities within the same tenant VXLAN network retain the same VXLAN segment ID, regardless of their location.

Fundamental Concepts of BGP EVPN VXLAN

This section provides information about the various fundamental concepts and terminologies that are involved in the working of BGP EVPN VXLAN.

VXLAN Overlay

An overlay network is a virtual network that is built over an existing Layer 2 or Layer 3 network by forming a static or dynamic tunnel that runs on top of the physical network infrastructure. The existing Layer 2 or Layer 3 network is what forms the underlay and is covered further below in this chapter.

When a data packet is sent through an overlay, the original packet or frame is packaged or encapsulated at a source edge device with an outer header and dispatched toward an appropriate destination edge device. The intermediate network devices forward the packet based on the outer header but are not aware of the data in the original packet. At the destination edge device, the packet is decapsulated by stripping off the overlay header and then forwarded based on the actual data within.

In the context of BGP EVPN VXLAN, VXLAN is used as the overlay technology to encapsulate the data packets and tunnel the traffic over a Layer 3 network. VXLAN creates a Layer 2 overlay network by using a MAC-in-UDP encapsulation. A VXLAN header is added to the original Layer 2 frame and it is then placed within a UDP-IP packet. A VXLAN overlay network is also called as a VXLAN segment. Only host devices and virtual machines within the same VXLAN segment can communicate with each other.

VXLAN Network Identifier

Each VXLAN segment is identified through a 24-bit segment ID, termed the VXLAN network identifier. This ensures that up to 16 million VXLAN segments can be present within the same administrative domain.

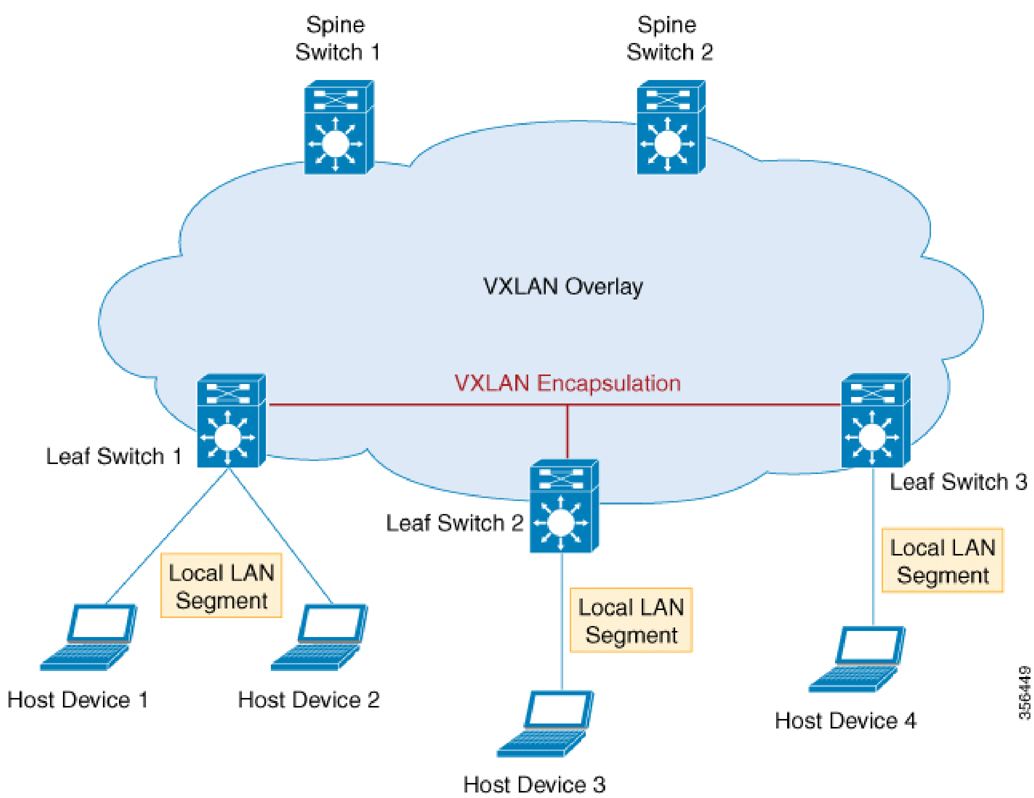
Virtual Tunnel End Points

Every VXLAN segment has tunnel edge devices known as Virtual Tunnel End points (VTEPs). These devices sit at the edge of the VXLAN network and are responsible for creating instances of VXLAN tunnels, and for performing VXLAN encapsulation and decapsulation.

A VTEP has a switch interface on the local LAN segment to support local endpoint communication through bridging, and an IP interface to interact with the transport IP network.

The IP interface has a unique IP address that identifies the VTEP on the transport IP network. The VTEP 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 following figure illustrates the working of an overlay VXLAN network connecting various VTEPs:



Overlay Multicast

Overlay multicast is the method by which a overlay network forwards multicast traffic between various VTEPs present in the network. Tenant Routed Multicast (TRM) provides a mechanism to efficiently forward multicast traffic in a VXLAN overlay network. TRM is a BGP-EVPN based solution that enables multicast routing between sources and receivers connected on VTEPs in VXLAN fabric.

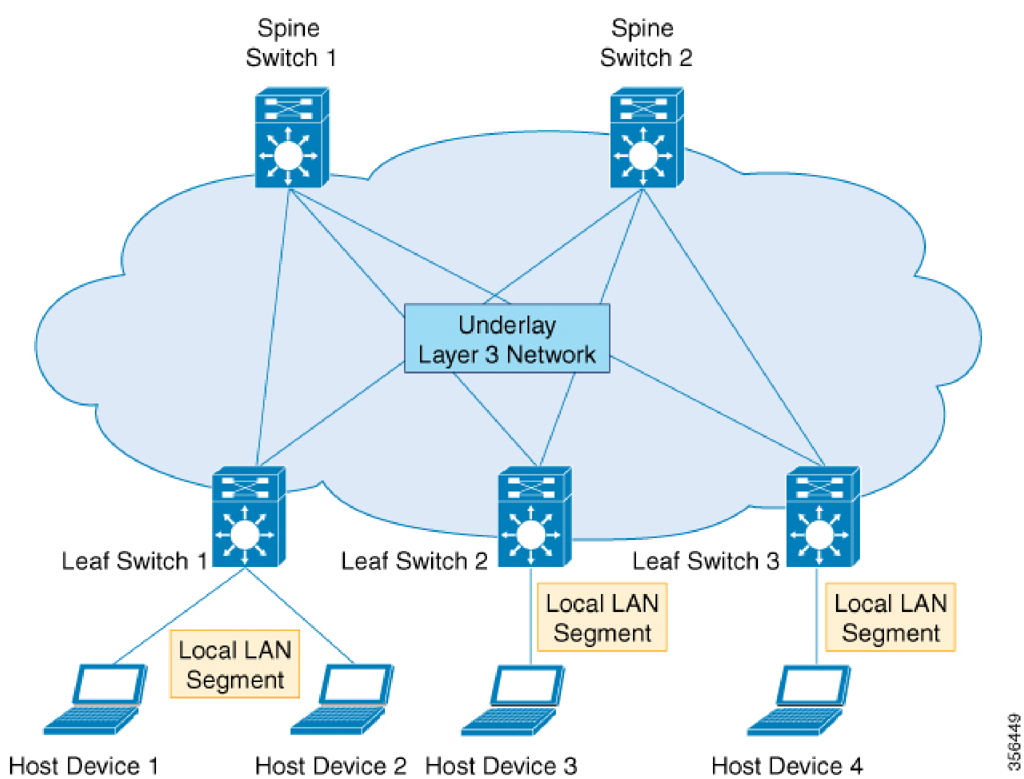
Without TRM, the multicast traffic is sent as part of the underlay network in the form of BUM traffic either using underlay multicast or ingress replication methods. This does not allow sources and receivers that are present across different subnets to communicate with each other. Using TRM, multicast communication is moved out of the BUM underlay traffic. This enables multicast communication in the overlay network irrespective of the subnet in which the source or the receiver resides.

Underlay

An underlay network is the physical network over which the virtual overlay network is established. Once the overlay network is defined along with the data-plane encapsulation, a method of transport is required to move the data across the physical network underneath. This method of transport is typically an underlay transport network, or simply the underlay.

In BGP EVPN VXLAN, the underlay Layer 3 network transports the VXLAN-encapsulated packets between the source and destination VTEPs and provides reachability between them. The VXLAN overlay and the underlying IP network between the VTEPs are independent of each other.

The following figure illustrates an underlay network:



EVPN Control Plane

The overlay requires a mechanism to know which end host device is behind which overlay edge device. VXLAN natively operates on a flood and learn mechanism where broadcast, unknown unicast and multicast

(BUM) traffic in a given VXLAN network is sent over the IP core to every VTEP that has membership in that network. IP multicast is used to send traffic over the network. The receiving VTEPs decapsulate the packet and, based on the inner frame, perform Layer 2 MAC learning. The inner source MAC address is learned against the outer source IP address corresponding to the source VTEP. In this way, reverse traffic is unicasted toward the previously learnt end host.

The drawback of the flood and learn mechanism is that it does not allow scalability in a VXLAN network. In order to address this issue, a control plane is used to manage the MAC address learning and VTEP discovery. In BGP EVPN VXLAN deployments, Ethernet Virtual Private Network (EVPN) is used as the control plane. EVPN control plane provides the capability to exchange both MAC address and IP address information. EVPN uses Multi Protocol Border Gateway Protocol (MP-BGP) as the routing protocol to distribute reachability information pertaining to the VXLAN overlay network, including endpoint MAC addresses, endpoint IP addresses, and subnet reachability information. BGP EVPN distribution protocol facilitates the mapping information to be built by the tunnel edge devices in the location-identity mapping database.

Route Target

A route target is an extended attribute in EVPN route updates that controls route distribution in a multi-tenant network. EVPN VTEPs have an import route target setting and an export route target setting for every VRF and Layer 2 Virtual Network Instance (VNI). When a VTEP advertises EVPN routes, it affixes its export route target in the route update. These routes are received by the other VTEPs in the network. The receiving VTEPs compare the route target value carried with the route against their own local import route target setting. If the two values match, the route is accepted and programmed in the routing table. Otherwise, the route is not imported.

EVPN Route Types

The EVPN control plane advertises the following types of information:

- Route type 1 – This is an Ethernet Auto-Discovery (EAD) route type used to advertise Ethernet segment identifier, Ethernet Tag ID, and EVPN instance information. EAD route advertisements may be sent for each EVPN instance or for each Ethernet segment.
- Route type 2 – This advertises endpoint reachability information, including MAC and IP addresses of the endpoints or VTEPs.
- Route type 3 – This performs multicast router advertisement, announcing the capability and intention to use ingress replication for specific VNIs.
- Route type 4 – This is an Ethernet Segment route used to advertise the Ethernet segment identifier, IP address length, and the originating router's IP address.
- Route type 5 – This is an IP prefix route used to advertise internal IP subnet and externally learned routes to a VXLAN network.

EVPN Instance

An EVPN Instance (EVI) represents a Virtual Private Network (VPN) on a VTEP. It is the equivalent of IP VRF in Layer 3 VPN and is also known as a MAC VRF.

Ethernet Segment

An Ethernet segment is associated with an access-facing interface of a VTEP and represents the connection with a host device. Each Ethernet segment is assigned a unique value known as Ethernet segment identifier

(ESI). When a host device is connected to more than one VTEPs, then the ESI for these connections remains the same.

EVPN Multihoming

EVPN multihoming allows you to connect a Layer 2 device or an end host device to more than one leaf switch in the VXLAN network. This provides redundancy and allows network optimization over single-homed topologies where the customer network is connected to a single leaf switch. Redundancy in the connection with the leaf switches ensures that there is no traffic disruption when there is a network failure. Multihomed topologies are more resilient, secure and efficient than single-homed topologies. EVPN multihoming operates in single-active and all-active redundancy modes.

Stretched VLAN and Subnet

By running over the existing networking infrastructure, EVPN VXLAN provides a means to stretch a Layer 2 network. EVPN VXLAN overlay allows Layer 2 segments and broadcast domains to be extended across sites or campus buildings over a Layer 3 core network. Layer 2 extension with EVPN VXLAN simplifies end user IP address management and provides seamless mobility in large campus networks.

Spine Leaf Architecture

Spine-leaf architecture is a two-layer network topology where one layer is composed of leaf switches and the other layer has one or more spine switches. This design connects all the leaf switches by providing multiple paths through the various spine switches.

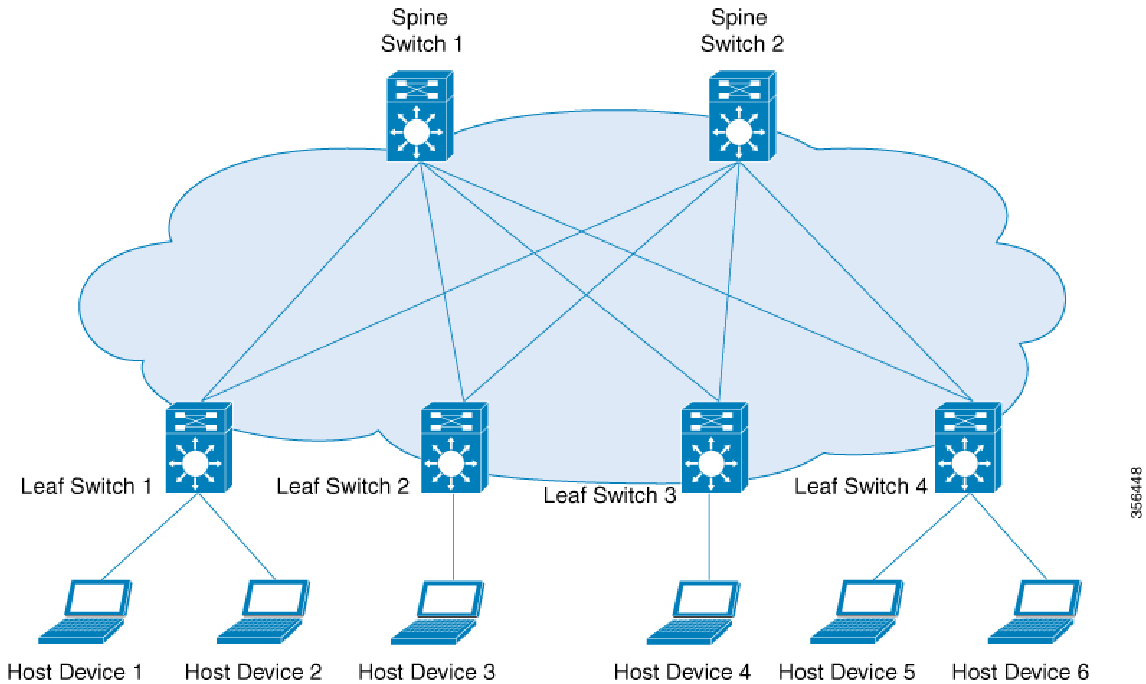
Spine Switch

Spine switches are the connecting nodes between all the leaf switches. They forward the traffic between the leaf switches and are unaware of the endpoint addresses. By providing multiple paths to connect the leaf switches, spine switches provide redundancy to the network.

Leaf Switch

Leaf switches are the nodes that are connected to the host or access devices. As a leaf switch sits on the edge of the network, it is also called as an edge or Network Virtualization Edge (NVE). When a host device on one leaf switch tries to communicate with a host device on another leaf switch, the traffic between the leaf switches is sent through a spine switch. Leaf switches function as VTEPs in a VXLAN network and perform the encapsulation and decapsulation.

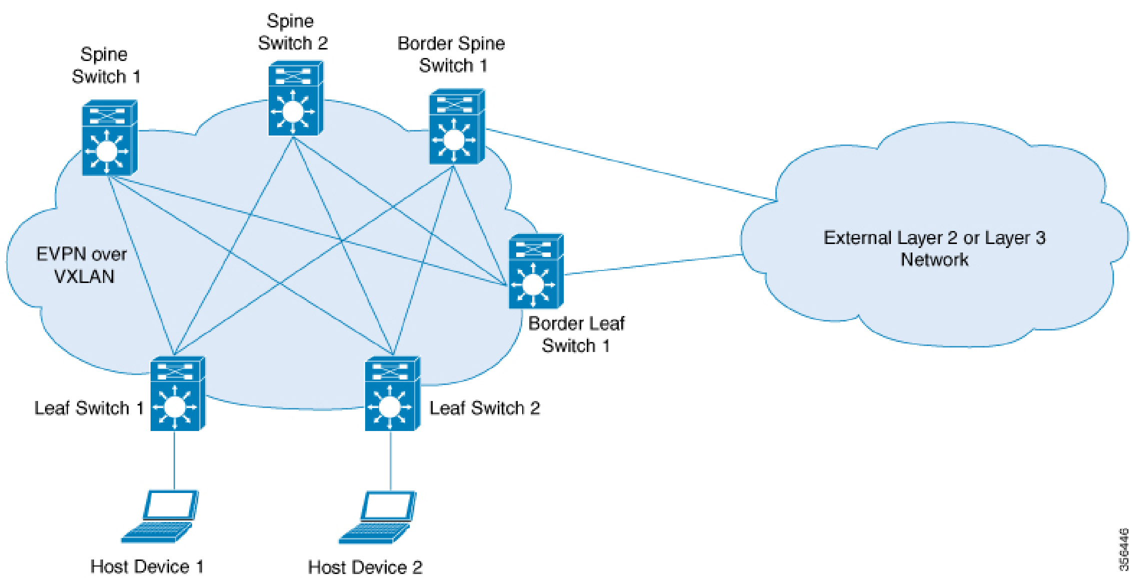
The following image shows a typical spine-leaf topology where four leaf switches are connected through two spine switches:



Border Spine Switch and Border Leaf Switch

External connectivity of the VXLAN fabric with other Layer 2 and Layer 3 networks is facilitated through nodes known as border nodes. If the border functionality is established through a spine switch, it is known as a border spine switch. If it is established through a leaf switch, it is known as a border leaf switch.

The following image shows a spine-leaf topology with one border leaf switch and one border spine switch connecting the fabric with an external network:



Integrated Routing and Bridging

EVPN VXLAN supports Integrated Routing and Bridging (IRB) functionality which allows the VTEPs in a VXLAN network to forward both Layer 2 (bridged) and Layer 3 (routed) traffic. When a VTEP forwards Layer 2 traffic, it is said to be performing bridging. Similarly, when a VTEP forwards Layer 3 traffic, it is said to be performing routing. The traffic between different subnets is forwarded through the VXLAN gateways. IRB is implemented in two ways:

- Asymmetric IRB
- Symmetric IRB

For more information about IRB, see Information About EVPN VXLAN Integrated Routing and Bridging, on page 51 section.

VXLAN Gateways

A VXLAN Gateway is an entity in the network that forwards traffic between VXLAN segments, or from a VXLAN environment to a non-VXLAN environment. Leaf switches in a VXLAN network can function as both Layer 2 and Layer 3 VXLAN gateways.

Layer 2 VXLAN gateways forward traffic within the same VLAN. Layer 2 VXLAN gateways allow VXLAN to VLAN bridging by mapping a VNI segment to a VLAN.

Layer 3 VXLAN gateways forward traffic to a different VLAN. Layer 3 VXLAN gateways allow both VXLAN to VXLAN routing as well as VXLAN to VLAN routing. VXLAN to VXLAN routing provides Layer 3 connectivity between two VNIs where as VXLAN to VLAN routing provides connectivity between a VNI and a VLAN.

Layer 2 Virtual Network Instance

The creation of a VXLAN overlay network allows host devices connected to various leaf nodes, that are separated by multiple Layer 3 networks, to interact as if they were connected to a single Layer 2 network, which is the VXLAN segment. This logical Layer 2 segment is called as Layer 2 VNI. The traffic that flows through a Layer 2 VNI between two VLANs within the same subnet is known as bridged traffic.

A VLAN that is locally defined on a VTEP can be mapped to a Layer 2 VNI. In order to allow host devices to connect to a Layer 2 VNI, the connected VLAN must be mapped to the Layer 2 VNI, and then the Layer 2 VNI is associated with the Network Virtualization Edge (NVE) logical interface on the VTEP.

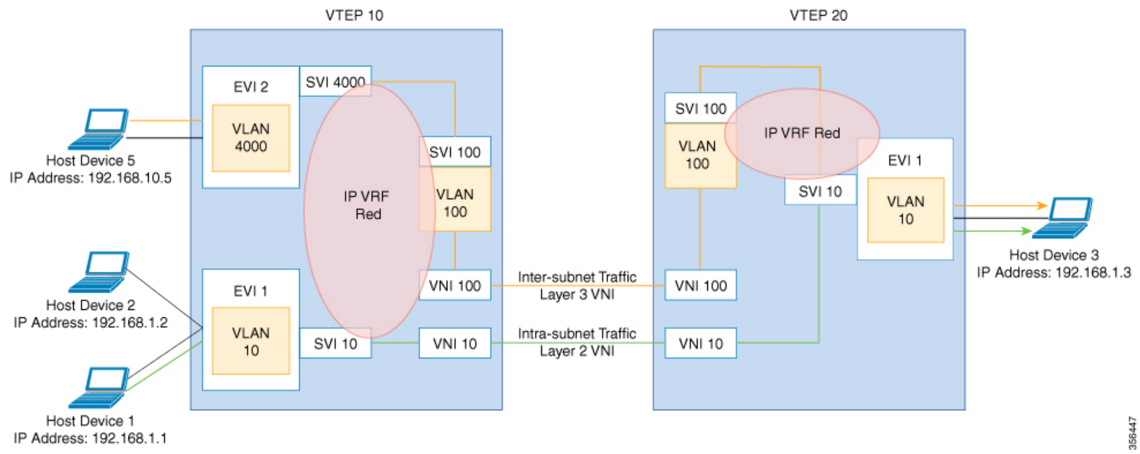
Layer 3 Virtual Network Instance

When endpoints connected to a Layer 2 VNI need to communicate with endpoints belonging to different IP subnets, they send the traffic to their default gateway. Communication between endpoints belonging to different Layer 2 VNIs is possible only through a Layer 3 routing function. In an EVPN VXLAN deployment, the various Layer 2 segments that are defined by combining the local VLANs and the global Layer 2 VNIs can be associated to a VRF in order to communicate.

A Layer 3 VNI facilitates Layer 3 segmentation for every VRF on a VTEP. This is done by mapping each VRF instance to a unique Layer 3 VNI in the network and associating the various Layer 2 VNIs for a VTEP to the same VRF. This allows inter- VXLAN communication throughout the Layer 3 VNI within a particular

VRF instance. The use of VRFs to enable a logical Layer 3 isolation is known as multi-tenancy. The traffic that flows through a Layer 3 VNI between two VLANs in different subnets is known as routed traffic.

The following image shows the movement of traffic between host devices in same and different subnets through Layer 2 and Layer 3 VNIs:



Mobility

The identity of an endpoint in the BGP EVPN control plane is derived from its MAC address and IP address, and BGP EVPN provides a mechanism to support endpoint mobility within a VXLAN overlay.

RFC 7432 defines the scope of endpoint mobility within the VXLAN fabric.

MAC Mobility and Duplicate MAC Detection

A MAC move occurs when an endpoint (or host) moves from one port to another. The new port may be within the same VTEP, or in a different VTEP, in the same VLAN. The BGP EVPN control plane resolves such moves by advertising MAC routes (EVPN route type 2). When an endpoint's MAC address is learned on a new port, the new VTEP it is in advertises (on the BGP EVPN control plane) that it is the local VTEP for the host. All other VTEPs receive the new MAC route.

A host may move several times, causing the corresponding VTEPs to advertise as many MAC routes. There may also be a delay between the time a new MAC route is advertised and when the old route is withdrawn from the route tables of other VTEPs, resulting in two locations briefly having the same MAC route. Here, a MAC mobility sequence number helps decide the most current of the MAC routes.

When the host MAC address is learned for the first time, the MAC mobility sequence number is set to 0. The value 0 indicates that the MAC address has not had a mobility event, and the host is still at the original location. If a MAC mobility event is detected, a new Route type 2 (MAC or IP advertisement) is added to the BGP EVPN control plane by the new VTEP below which the endpoint moved (its new location). Every time the host moves, the VTEP that detects its new location increments the sequence number by 1 and then advertises the MAC route for that host on the BGP EVPN control plane. On receiving the MAC route at the old location (VTEP), the old VTEP withdraws the old route.

A case may arise in which the same MAC address is simultaneously learned on two different ports. The EVPN control plane detects this condition and alerts the user that there is a duplicate MAC. The duplicate MAC condition may be cleared either by manual intervention, or automatically when the MAC address ages out on one of the ports.

IP Mobility and Duplicate IP Detection

BGP EVPN supports IP mobility in a similar manner to the way it supports MAC mobility. The principal difference is that an IP move is detected when the IP address is learned on a different MAC address, regardless of whether it was learned on the same port or a different port. A duplicate IP address is detected when the same IP address is simultaneously learned on two different MAC addresses, and the user is alerted when this occurs.

CHAPTER 2

Configuring EVPN VXLAN Layer 2 Overlay Network

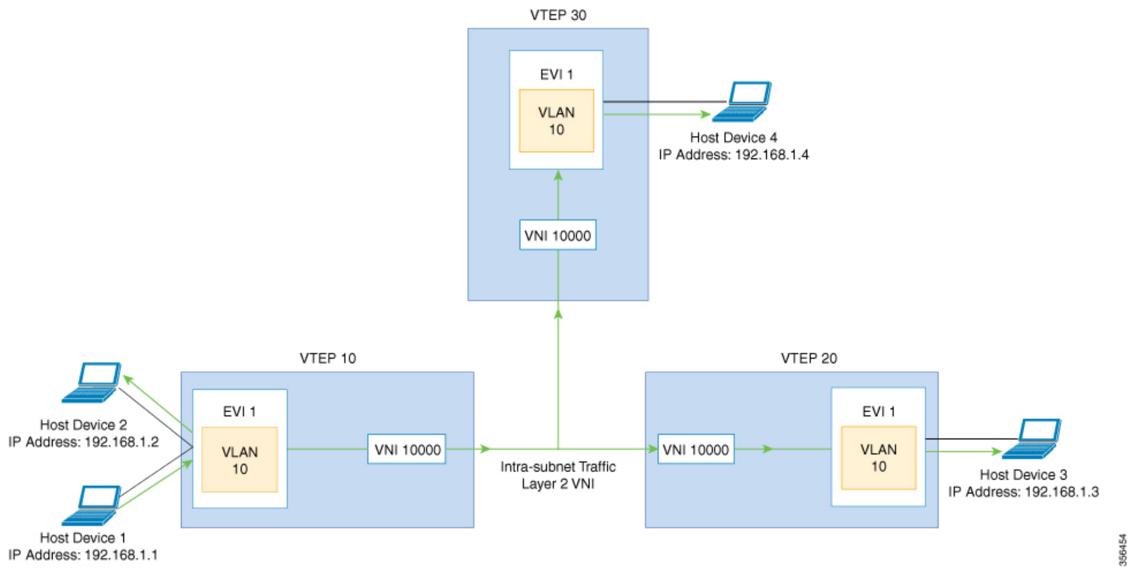
Information About EVPN VXLAN Layer 2 Overlay Network

An EVPN VXLAN Layer 2 overlay network allows host devices in the same subnet to send bridged or Layer 2 traffic to each other. The network forwards the bridged traffic using a Layer 2 virtual network instance (VNI).

Broadcast, Unknown Unicast, and Multicast Traffic

Multidestination Layer 2 traffic in a VXLAN network is typically referred to as broadcast, unknown unicast, and multicast (BUM) traffic. In a BGP EVPN VXLAN fabric, the underlay network forwards the BUM traffic to all the endpoints connected to a common Layer 2 broadcast domain in the VXLAN overlay.

The following image shows the flow of BUM traffic through a Layer 2 VNI. The network forwards BUM traffic from host device 1 to all the VTEPs which in turn send the traffic to all the host devices in the same subnet.



The MP-BGP EVPN control plane uses two different methods to forward BUM traffic in a VXLAN network:

- Underlay Multicast
- Ingress Replication

Underlay Multicast

In underlay multicast, the underlay network replicates the traffic through a multicast group. Forwarding BUM traffic using underlay multicast requires the configuration of IP multicast in the underlay network. A single copy of the BUM traffic moves from the ingress or source VTEP towards the underlay transport network. The network forwards this copy along the multicast tree so that it reaches all egress or destination VTEPs participating in the given multicast group. Various branch points in the network replicate the copy as it travels along the multicast tree. The branch points replicate the copy only if the receivers are part of the multicast group associated with the VNI.

BUM traffic forwarding through underlay multicast is achieved by mapping a Layer 2 VNI to the multicast group. This mapping must be configured on all the VTEPs associated with the Layer 2 VNI. When a VTEP joins the multicast group, it receives all the traffic that is forwarded on that group. If the VTEP receives traffic in a VNI that is not associated with it, it simply drops the traffic. This approach maintains a single link within the network, thus providing an efficient way to forward BUM traffic.

Ingress Replication

Ingress replication, or headend replication, is a unicast approach to handle multidestination Layer 2 overlay BUM traffic. Ingress replication involves an ingress device replicating every incoming BUM packet and sending them as a separate unicast to the remote egress devices. Ingress replication happens through EVPN route type 3, also called as inclusive multicast ethernet tag (IMET) route. BGP EVPN ingress replication uses IMET route for auto-discovery of remote peers in order to set up the BUM tunnels over VXLAN. Using ingress replication to handle BUM traffic can result in scaling issues as an ingress device needs to replicate the BUM traffic as many times as there are VTEPs associated with the Layer 2 VNI.

Ingress Replication Operation

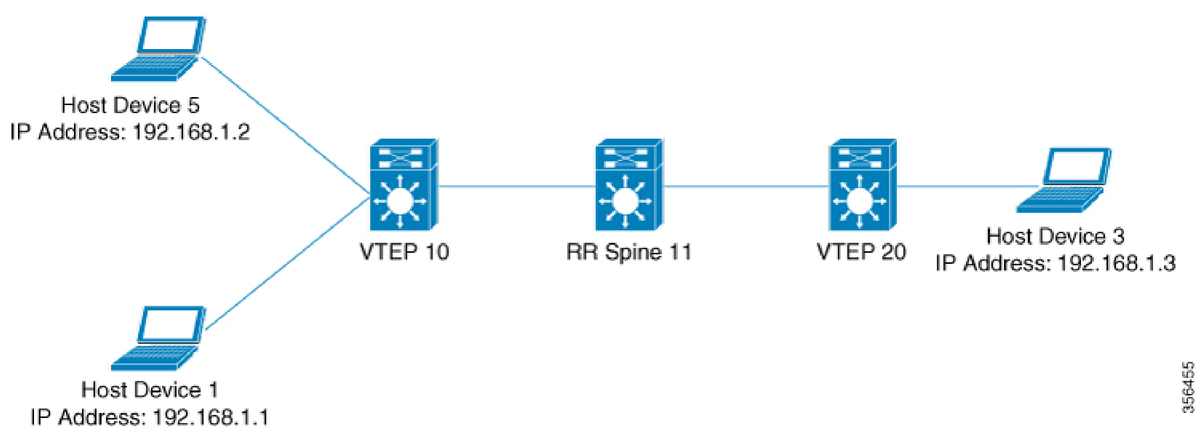
IMET routes carry the remote or egress VNIs advertised from the remote peers, which can be different from the local VNI. The network creates a VXLAN tunnel adjacency when an ingress device receives IMET ingress replication routes from remote NVE peers. The tunnel adjacency is a midchain adjacency which contains IP or UDP encapsulation for the VXLAN Tunnel. If there is more than one VNI along the tunnel, then multiple VNIs share the tunnel. Ingress replication on EVPN can have multiple unicast tunnel adjacencies and different egress VNIs for each remote peer.

The network builds a flooded replication list with the routes advertised by each VTEP. The dynamic replication list stores all the remote destination peers discovered on a BGP IMET route in the same Layer 2 VNI. The replication list gets updated every time you configure the Layer 2 VNI at a remote peer. The network removes the tunnel adjacency and VXLAN encapsulation from the replication list every time a remote NVE peer withdraws the IMET ingress replication route. The network deletes the tunnel adjacency when there is no NVE peer using it.

Any BUM traffic that reaches the ingress device gets replicated after the replication list is built. The ingress device forwards the replicated traffic throughout the network to all the remote peers in the same VNI.

How to Configure EVPN VXLAN Layer 2 Overlay Network

The following figure shows a sample topology of an EVPN VXLAN Network. Host device 1 and host device 3 are part of the same subnet. The network forwards BUM traffic from host device 1 to host device 3 using a Layer 2 VNI through either underlay multicast or ingress replication methods.



Note

In a two-VTEP topology, a spine switch is not mandatory. For information about configuration of spine switches in an EVPN VXLAN network, see *Configuring Spine Switches in a BGP EVPN VXLAN Fabric* module.

Perform the following set of procedures to configure an EVPN VXLAN Layer 2 overlay network and forward the BUM traffic:

- Configure Layer 2 VPN EVPN on the VTEPs.
- Configure an EVPN instance in the VLAN on the VTEPs.

- Configure the access-facing interface in the VLAN on the VTEPs.
- Configure the loopback interface on the VTEPs.
- Configure the network virtualization endpoint (NVE) interface on the VTEPs.
- Configure BGP with EVPN address family on the VTEPs.
- Configure underlay multicast, if the specified replication type is static. For more information, see *IP Multicast Routing Configuration Guide*.

Configuring Layer 2 VPN EVPN on a VTEP

To configure the Layer 2 VPN EVPN parameters on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2vpn evpn** **Example:** `Device(config)# l2vpn evpn`	Enters EVPN configuration mode.
Step 4	**replication-type** {**ingress** | **static**} **Example:** `Device(config-evpn)# replication-type static`	Configures the Layer 2 VPN EVPN replication type. **Note** Configure the Layer 2 VPN EVPN replication type as static, if multicast is enabled in the underlay network for EVPN BUM traffic. When the Layer 2 VPN EVPN replication type is configured as static, the IMET route is not advertised and forwarding of BUM traffic relies on underlay multicast being configured on each VTEP.
Step 5	**router-id** *loopback-interface-id* **Example:** `Device(config-evpn)# router-id loopback 0`	Specifies the interface that will supply the IP addresses to be used in auto-generating route distinguishers.

	Command or Action	Purpose
Step 6	**default-gateway advertise** **Example:** `Device(config-evpn)# default-gateway advertise`	(Optional) Enables default gateway advertisement on the switch. To configure distributed anycast gateway in a VXLAN network using MAC aliasing, enable default gateway advertisement on all the leaf switches in the network. This command is applicable in integrated routing and bridging (IRB) scenarios where Layer 2 and Layer 3 VNIs coexist in a VRF. Refer to *Configuring EVPN VXLAN Integrated Routing and Bridging* module for more details. This command is mandatory only if the same MAC address is not manually configured on all the access SVIs. **Note** Use the **default-gateway advertise** {**enable** | **disable**} command in EVPN instnace configuration mode to override the global default gateway advertisement settings and enable or disable it for a specific EVPN instance.
Step 7	**logging peer state** **Example:** `Device(config-evpn)# logging peer state`	(Optional) Displays syslog message when the first route is received or the last route is withdrawn from a given remote VTEP.
Step 8	**mac duplication limit** *limit-number* **time** *time-limit* **Example:** `Device(config-evpn)# mac duplication limit 20 time 5`	(Optional) Changes parameters for detecting duplicate MAC addresses.
Step 9	**ip duplication limit** *limit-number* **time** *time-limit* **Example:** `Device(config-evpn)# ip duplication limit 20 time 5`	(Optional) Changes parameters for detecting duplicate IP addresses.
Step 10	**route-target auto vni** **Example:** `Device(config-evpn)# route-target auto vni`	(Optional) Specifies to use VNI instead of EVPN instance number to auto-generate route target.
Step 11	**exit** **Example:** `Device(config-evpn)# exit`	Exits EVPN configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 12	**l2vpn evpn instance** *evpn-instance-number* **vlan-based** **Example:** `Device(config)# l2vpn evpn instance 1 vlan-based`	Configures a VLAN based EVPN instance in Layer 2 VPN configuration mode. An EVPN instance needs to be explicitly configured only when something needs to be configured per EVPN instance such as a route target.
Step 13	**encapsulation vxlan** **Example:** `Device(config-evpn-evi)# encapsulation vxlan`	(Optional) Defines the encapsulation format as VXLAN. The encapsulation format is VXLAN by default.
Step 14	**replication-type** {**ingress** | **static**} **Example:** `Device(config-evpn-evi)# replication-type ingress`	(Optional) Sets the replication type for the EVPN instance. In case a global replication type has already been configured, this overrides the global setting.
Step 15	**default-gateway advertise** {**enable** | **disable**} **Example:** `Device(config-evpn-evi)# default-gateway advertise disable`	(Optional) Enables or disables the default gateway advertisement for the EVPN instance. In case default gateway advertisement has already been globally configured, this overrides the global setting. This command is mandatory only if the same MAC address is not manually configured on all the access SVIs. To configure distributed anycast gateway in a VXLAN network using MAC aliasing, enable default gateway advertisement on all the leaf switches in the network.
Step 16	**ip local-learning** {**enable** | **disable**} **Example:** `Device(config-evpn-evi)# ip local-learning disable`	(Optional) Enables or disables local IP address learning for the specified EVPN instance. In case IP address learning has already been globally configured, this overrides the global setting.
Step 17	**no auto-route-target** **Example:** `Device(config-evpn-evi)# no auto-route-target`	(Optional) Disables auto generation of route targets.
Step 18	**rd** *rd-value* **Example:** `Device(config-evpn-evi)# rd 65000:100`	(Optional) Configures a route distinguisher manually.

	Command or Action	Purpose
Step 19	**route-target** {**import** | **export** | **both**} *rt-value* **Example:** `Device(config-evpn-evi)# route-target both 65000:100`	(Optional) Configures route targets manually. **Note** Configure route targets manually if the auto-generated route target values (ASN:EVI or ASN:VNI) are different between the VTEPs.
Step 20	**end** **Example:** `Device(config-evpn-evi)# end`	Returns to privileged EXEC mode.

Configuring an EVPN Instance on the VLAN on a VTEP

To configure an EVPN instance on the VLAN on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vlan configuration** *vlan-id* **Example:** `Device(config)# vlan configuration 11`	Enters VLAN feature configuration mode for the specified VLAN interface.
Step 4	**member evpn-instance** *evpn-instance-id* **vni** *l2-vni-number* **Example:** `Device(config-vlan)# member evpn-instance 1 vni 10000`	Adds EVPN instance as a member of the VLAN configuration. The VNI here is used as a Layer 2 VNI.
Step 5	**end** **Example:** `Device(config-vlan)# end`	Returns to privileged EXEC mode.

Configuring the Access-Facing Interface in the VLAN on a VTEP

To configure the access-facing interface in the VLAN on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-name* **Example:** `Device(config)# interface GigabitEthernet1/0/1`	Enters interface configuration mode for the specified interface.
Step 4	**switchport access vlan** *vlan-id* **Example:** `Device(config-if)# switchport access vlan 11`	Configures the interface as a static-access port of the specified VLAN. Interface can also be configured as a trunk interface, if required.
Step 5	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring the Loopback Interface on a VTEP

To configure the loopback interface on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *loopback-interface-id* **Example:** `Device(config)# interface Loopback0`	Enters interface configuration mode for the specified Loopback interface.

	Command or Action	Purpose
Step 4	**ip address** *ipv4-address* **Example:** `Device(config-if)# ip address 10.12.11.11`	Configures the IP address for the Loopback interface.
Step 5	**ip pim sparse mode** **Example:** `Device(config-if)# ip pim sparse mode`	Enables Protocol Independent Multicast (PIM) sparse mode on the Loopback interface.
Step 6	**end** **Example:** `Device(config-vlan)# end`	Returns to privileged EXEC mode.

Configuring the NVE Interface on a VTEP

To add a VNI member to the NVE interface of a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *nve-interface-id* **Example:** `Device(config)# interface nve1`	Defines the interface to be configured as a trunk, and enters interface configuration mode.
Step 4	**no ip address** **Example:** `Device(config-if)# no ip address`	Disables IP processing on the interface by removing its IP address.
Step 5	**source-interface** *loopback-interface-id* **Example:** `Device(config-if)# source-interface loopback0`	Sets the IP address of the specified loopback interface as the source IP address.
Step 6	**host-reachability protocol bgp** **Example:** `Device(config-if)# host-reachability protocol bgp`	Configures BGP as the host-reachability protocol on the interface.

	Command or Action	Purpose
Step 7	**member vni** *layer2-vni-id* {**ingress-replication** | **mcast-group** *multicast-group-address* **Example:** `Device(config-if)# member vni 10000 mcast-group 227.0.0.1`	Associates the Layer 2 VNI member with the NVE. The specified replication type must match the replication type that is configured globally or for the specific EVPN instance. Use **mcast-group** keyword for static replication and **ingress-replication** keyword for ingress replication.
Step 8	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring BGP on a VTEP with EVPN Address Family

To configure BGP on a VTEP with EVPN address family and with spine switch as the neighbor, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 1`	Enables a BGP routing process, assigns it an autonomous system number, and enters router configuration mode.
Step 4	**bgp log-neighbor-changes** **Example:** `Device(config-router)# bgp log-neighbor-changes`	(Optional) Enables the generation of logging messages when the status of a BGP neighbor changes. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.
Step 5	**bgp update-delay** *time-period* **Example:** `Device(config-router)# bgp update-delay 1`	(Optional) Sets the maximum initial delay period before sending the first update. The range is 1 to 3600 seconds. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.

	Command or Action	Purpose
Step 6	**bgp graceful-restart** **Example:** `Device(config-router)# bgp graceful-restart`	(Optional) Enables the BGP graceful restart capability for all BGP neighbors. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.
Step 7	**no bgp default ipv4-unicast** **Example:** `Device(config-router)# no bgp default ipv4-unicast`	(Optional) Disables default IPv4 unicast address family for BGP peering session establishment. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.
Step 8	**neighbor** *ip-address* **remote-as** *number* **Example:** `Device(config-router)# neighbor 10.11.11.11 remote-as 1`	Defines multiprotocol-BGP neighbors. Under each neighbor, define the Layer 2 Virtual Private Network (L2VPN) EVPN configuration. Use the IP address of the spine switch as the neighbor IP address.
Step 9	**neighbor** {*ip-address* | *group-name*} **update-source** *interface* **Example:** `Device(config-router)# neighbor 10.11.11.11 update-source Loopback0`	Configures update source. Update source can be configured per neighbor or per peer-group. Use the IP address of the spine switch as the neighbor IP address.
Step 10	**address-family l2vpn evpn** **Example:** `Device(config-router)# address-family l2vpn evpn`	Specifies the L2VPN address family and enters address family configuration mode.
Step 11	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 10.11.11.11 activate`	Enables the exchange information from a BGP neighbor. Use the IP address of the spine switch as the neighbor IP address.
Step 12	**neighbor** *ip-address* **send-community** [**both** | **extended** | **standard**] **Example:** `Device(config-router-af)# neighbor 10.11.11.11 send-community both`	Specifies the communities attribute sent to a BGP neighbor. Use the IP address of the spine switch as the neighbor IP address.
Step 13	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 14	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-router)# end`	

Configuration Examples for EVPN VXLAN Layer 2 Overlay Network

This section provides an example for configuring an EVPN VXLAN Layer 2 overlay network. This example shows a sample configuration for a VXLAN network with 2 VTEPs, VTEP 1 and VTEP 2, connected to perform bridging.

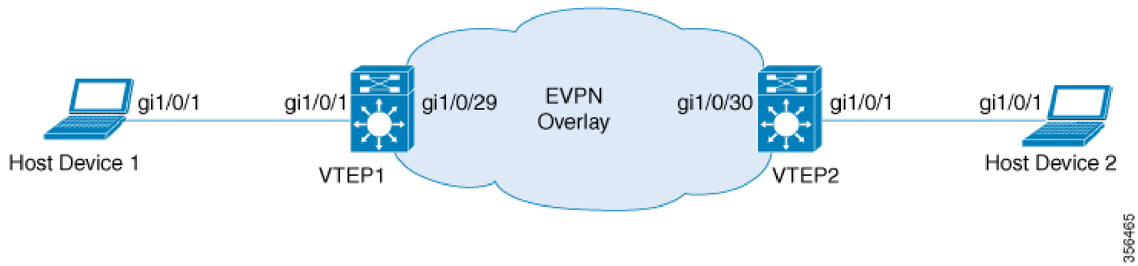


Table 1: Configuration Example for a VXLAN Network with Two VTEPs Connected to Perform Bridging

VTEP 1	VTEP 2

<table>
<tr><th>VTEP 1</th><th>VTEP 2</th></tr>
<tr><td><pre>VTEP1# show running-config
Building configuration...
!
hostname VTEP1
!
ip routing
ip multicast-routing
!
l2vpn evpn
 replication-type static
 router-id Loopback0
!
l2vpn evpn instance 1 vlan-based
 encapsulation vxlan
 route-target export 103:1
 route-target import 104:1
!
system mtu 9150
!
vlan configuration 201
 member evpn-instance 1 vni 6000
!
!
interface Loopback0
 ip address 10.1.1.10 255.255.255.255
 ip pim sparse-mode
!
!
interface GigabitEthernet1/0/1
description host1-interface
 switchport access vlan 201
 switchport mode access
!
!
interface GigabitEthernet1/0/29
 description core-underlay-interface
 no switchport
ip address 172.16.1.29 255.255.255.0
 ip pim sparse-mode
!
!
interface nve10
 no ip address
 source-interface Loopback0
 host-reachability protocol bgp
 member vni 6000 mcast-group 232.1.1.1
!
router ospf 1
 router-id 10.1.1.10
 network 10.1.1.0 0.0.0.255 area 0
 network 172.16.1.0 0.0.0.255 area 0
!
router bgp 10
 bgp router-id interface Loopback0
 bgp log-neighbor-changes
 bgp update-delay 1
 no bgp default ipv4-unicast
 neighbor 10.2.2.20 remote-as 10
 neighbor 10.2.2.20 update-source Loopback0
 !
 address-family ipv4
 exit-address-family</pre></td><td><pre>VTEP2# show running-config
Building configuration...
!
hostname VTEP2
!
ip routing
ip multicast-routing
!
l2vpn evpn
 replication-type static
 router-id Loopback0
!
l2vpn evpn instance 1 vlan-based
 encapsulation vxlan
 route-target export 104:1
 route-target import 103:1
!
system mtu 9150
!
vlan configuration 201
 member evpn-instance 1 vni 6000
!
!
interface Loopback0
 ip address 10.2.2.20 255.255.255.255
 ip pim sparse-mode
!
!
interface GigabitEthernet1/0/1
description host2-interface
 switchport access vlan 201
 switchport mode access
!
!
interface GigabitEthernet1/0/30
 description core-underlay-interface
 no switchport
 ip address 172.16.1.30 255.255.255.0
 ip pim sparse-mode
!
!
interface nve10
 no ip address
 source-interface Loopback0
 host-reachability protocol bgp
 member vni 6000 mcast-group 232.1.1.1
!
router ospf 1
 router-id 10.2.2.20
 network 10.2.2.0 0.0.0.255 area 0
 network 172.16.1.0 0.0.0.255 area 0
!
router bgp 10
 bgp router-id interface Loopback0
 bgp log-neighbor-changes
 bgp update-delay 1
 no bgp default ipv4-unicast
 neighbor 10.1.1.10 remote-as 10
 neighbor 10.1.1.10 update-source Loopback0
 !
 address-family ipv4
 exit-address-family</pre></td></tr>
</table>

<table>
<tr><th>VTEP 1</th><th>VTEP 2</th></tr>
<tr><td><pre>
 !
 address-family l2vpn evpn
  neighbor 10.2.2.20 activate
  neighbor 10.2.2.20 send-community both
 exit-address-family
 !
ip pim rp-address 10.1.1.10
!
end
</pre></td><td><pre>
 !
 address-family l2vpn evpn
  neighbor 10.1.1.10 activate
  neighbor 10.1.1.10 send-community both
 exit-address-family
!
ip pim rp-address 10.1.1.10
!
end
</pre></td></tr>
</table>

The following examples provide outputs for **show** commands on VTEP 1 and VTEP 2 in the topology configured above.

- show l2vpn evpn peers vxlan, on page 25
- show nve peers, on page 25
- show l2vpn evpn mac, on page 26
- show bgp l2vpn evpn all, on page 26
- show platform software fed switch active matm macTable vlan, on page 27

show l2vpn evpn peers vxlan

VTEP 1

This example shows the output for the **show l2vpn evpn peers vxlan** command on VTEP 1:

```
VTEP1# show l2vpn evpn peers vxlan
Interface VNI      Peer-IP                  Num routes eVNI     UP time
--------- -------- ------------------------ ---------- -------- --------
nve10     6000     10.2.2.20                3          6000     00:12:44
```

VTEP 2

This example shows the output for the **show l2vpn evpn peers vxlan** command on VTEP 2:

```
VTEP2# show l2vpn evpn peers vxlan
Interface VNI      Peer-IP                  Num routes eVNI     UP time
--------- -------- ------------------------ ---------- -------- --------
nve10     6000     10.1.1.10                3          6000     00:01:41
```

show nve peers

VTEP 1

This example shows the output for the **show nve peers** command on VTEP 1:

```
VTEP1# show nve peers
Interface  VNI      Type Peer-IP          RMAC/Num_RTs   eVNI     state flags UP time
nve10      6000     L2CP 10.2.2.20        3              6000       UP   N/A  00:12:48
```

VTEP 2

This example shows the output for the **show nve peers** command on VTEP 2:

```
VTEP2# show nve peers
Interface  VNI      Type Peer-IP          RMAC/Num_RTs   eVNI     state flags UP time
nve10      6000     L2CP 10.1.1.10        3              6000       UP   N/A  00:01:46
```

show l2vpn evpn mac

VTEP 1

This example shows the output for the **show l2vpn evpn mac** command on VTEP 1:

```
VTEP1# show l2vpn evpn mac
MAC Address    EVI   VLAN  ESI                      Ether Tag  Next Hop(s)
-------------- ----- ----- ------------------------ ---------- ---------------
0018.736c.5681 1     201   0000.0000.0000.0000.0000 0          10.2.2.20
0018.736c.56c3 1     201   0000.0000.0000.0000.0000 0          10.2.2.20
0059.dc50.ae01 1     201   0000.0000.0000.0000.0000 0          Gi1/0/1:201
0059.dc50.ae4c 1     201   0000.0000.0000.0000.0000 0          Gi1/0/1:201
```

VTEP 2

This example shows the output for the **show l2vpn evpn mac** command on VTEP 2:

```
VTEP2# show l2vpn evpn mac
MAC Address    EVI   VLAN  ESI                      Ether Tag  Next Hop(s)
-------------- ----- ----- ------------------------ ---------- ---------------
0018.736c.5681 1     201   0000.0000.0000.0000.0000 0          Gi1/0/1:201
0018.736c.56c3 1     201   0000.0000.0000.0000.0000 0          Gi1/0/1:201
0059.dc50.ae01 1     201   0000.0000.0000.0000.0000 0          10.1.1.10
0059.dc50.ae4c 1     201   0000.0000.0000.0000.0000 0          10.1.1.10
```

show bgp l2vpn evpn all

VTEP 1

This example shows the output for the **show bgp l2vpn evpn all** command on VTEP 1:

```
VTEP1# show bgp l2vpn evpn all
BGP table version is 101, local router ID is 10.1.1.10
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path, L long-lived-stale,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 10.1.1.10:1
 *>i  [2][10.1.1.10:1][0][48][0018736C5681][0][*]/20
                      10.2.2.20                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][0018736C56C3][0][*]/20
                      10.2.2.20                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][0018736C56C3][32][192.168.1.89]/24
```

```
                     10.2.2.20                0    100      0 ?
 *>   [2][10.1.1.10:1][0][48][0059DC50AE01][0][*]/20
                     ::                                 32768 ?
 *>   [2][10.1.1.10:1][0][48][0059DC50AE4C][0][*]/20
                     ::                                 32768 ?
 *>   [2][10.1.1.10:1][0][48][0059DC50AE4C][32][192.168.1.81]/24
                     ::                                 32768 ?
Route Distinguisher: 10.2.2.20:1
 *>i  [2][10.2.2.20:1][0][48][0018736C5681][0][*]/20
                     10.2.2.20                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][0018736C56C3][0][*]/20
                     10.2.2.20                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][0018736C56C3][32][192.168.1.89]/24
                     10.2.2.20                0    100      0 ?
```

VTEP 2

This example shows the output for the **show bgp l2vpn evpn all** command on VTEP 2:

```
VTEP2# show bgp l2vpn evpn all
BGP table version is 99, local router ID is 10.2.2.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path, L long-lived-stale,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 10.1.1.10:1
 *>i  [2][10.1.1.10:1][0][48][0059DC50AE01][0][*]/20
                     10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][0059DC50AE4C][0][*]/20
                     10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][0059DC50AE4C][32][192.168.1.81]/24
                     10.1.1.10                0    100      0 ?
Route Distinguisher: 10.2.2.20:1
 *>   [2][10.2.2.20:1][0][48][0018736C5681][0][*]/20
                     ::                                 32768 ?
 *>   [2][10.2.2.20:1][0][48][0018736C56C3][0][*]/20
                     ::                                 32768 ?
 *>   [2][10.2.2.20:1][0][48][0018736C56C3][32][192.168.1.89]/24
     Network          Next Hop            Metric LocPrf Weight Path
                     ::                                 32768 ?
 *>i  [2][10.2.2.20:1][0][48][0059DC50AE01][0][*]/20
                     10.1.1.10                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][0059DC50AE4C][0][*]/20
                     10.1.1.10                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][0059DC50AE4C][32][192.168.1.81]/24
                     10.1.1.10                0    100      0 ?
```

show platform software fed switch active matm macTable vlan

VTEP 1

This example shows the output for the **show platform software fed switch active matm mactable vlan** command on VTEP 1:

```
VTEP1# show platform software fed switch active matm macTable vlan 201
VLAN   MAC                  Type  Seq#   EC_Bi  Flags  machandle           siHandle
      riHandle            diHandle              *a_time  *e_time  ports
```

```
------------------------------------------------------------------------------------------
201    0018.736c.5681   0x1000001      0      0     64  0x7f5d852abaf8      0x7f5d850c1858
      0x7f5d8527def8      0x0                           0        0  RLOC 10.2.2.20 adj_id 81

201    0018.736c.56c3   0x1000001      0      0     64  0x7f5d855be2b8      0x7f5d850c1858
      0x7f5d8527def8      0x0                           0        0  RLOC 10.2.2.20 adj_id 81

201    0059.dc50.ae01         0x1     22      0      0  0x7f5d855c6388      0x7f5d85035248
      0x0                0x7f5d8517eae8           300       11  GigabitEthernet1/0/1

201    0059.dc50.ae4c         0x1     26      0      0  0x7f5d84fba3c8      0x7f5d85035248
      0x0                0x7f5d8517eae8           300       58  GigabitEthernet1/0/1


Total Mac number of addresses:: 4
```

VTEP 2

This example shows the output for the **show platform software fed switch active matm mactable vlan** command on VTEP 2:

```
VTEP2# show platform software fed switch active matm macTable vlan 201
VLAN   MAC                   Type  Seq#   EC_Bi  Flags  machandle           siHandle
      riHandle            diHandle              *a_time  *e_time  ports
------------------------------------------------------------------------------------------
201    0018.736c.5681         0x1     38      0      0  0x7f40e196cac8      0x7f40e196cf28
      0x0                0x7f40e0f6da38           300       12  GigabitEthernet1/0/1

201    0018.736c.56c3         0x1     39      0      0  0x7f40e19b6878      0x7f40e196cf28
      0x0                0x7f40e0f6da38           300       17  GigabitEthernet1/0/1

201    0059.dc50.ae01   0x1000001      0      0     64  0x7f40e19b88f8      0x7f40e1937b88
      0x7f40e193bd58      0x0                           0       17  RLOC 10.1.1.10 adj_id 28

201    0059.dc50.ae4c   0x1000001      0      0     64  0x7f40e194d638      0x7f40e1937b88
      0x7f40e193bd58      0x0                           0       17  RLOC 10.1.1.10 adj_id 28


Total Mac number of addresses:: 4
```

Verifying EVPN VXLAN Layer 2 Overlay Network

The following table lists the **show** commands that are used to verify a Layer 2 VXLAN overlay network:

Table 2: Commands to Verify EVPN VXLAN Layer 2 Overlay Network

Command	Purpose
show l2vpn evpn evi [**detail**]	Displays detailed information for a particular EVPN instance or all EVPN instances.
show l2vpn evpn mac [**detail**]	Displays the MAC address database for Layer 2 EVPN.
show l2vpn evpn mac ip [**detail**]	Displays the IP address database for Layer 2 EVPN.

Command	Purpose
show l2vpn evpn summary	Displays a summary of Layer 2 EVPN information.
show l2vpn evpn capabilities	Displays platform capability information for Layer 2 EVPN.
show l2vpn evpn peers	Displays Layer 2 EVPN peer route counts and up time.
show l2vpn evpn route-target	Displays Layer 2 EVPN import route targets.
show l2vpn evpn memory	Displays Layer 2 EVPN memory usage.
show l2route evpn summary	Displays a summary of EVPN routes.
show l2route evpn mac [**detail**]	Displays MAC address information learnt by the switch in the EVPN control plane.
show l2route evpn mac ip [**detail**]	Displays MAC and IP address information learnt by the switch in the EVPN control plane.
show l2route evpn imet detail	Displays the IMET route details for Layer 2 EVPN address family. This command shows details only about traffic forwarded using ingress replication.
show bgp l2vpn evpn	Displays BGP information for Layer 2 VPN EVPN address family.
show bgp l2vpn evpn route-type 2	Displays BGP information for route type 2 of L2VPN EVPN address family.
show bgp l2vpn evpn evi context	Displays context information for Layer 2 EVPN instances.
show bgp l2vpn evpn evi *evpn-instance-id* **route-type 3**	Displays route type 3 information for the specified Layer 2 EVPN instance. This command shows details only about traffic forwarded using ingress replication.
show l2fib bridge-domain *bridge-domain-number* **detail**	Displays detailed information for a Layer 2 forwarding information base bridge domain.
show l2fib bridge-domain *bridge-domain-number* **address unicast**	Displays unicast MAC address information for a Layer 2 forwarding information base bridge domain.
show nve vni	Displays information about VXLAN network identifier members associated with an NVE interface.
show nve vni *vni-id* **detail**	Displays detailed NVE interface state information for a VXLAN network identifier member.

Command	Purpose
show nve peers	Displays NVE interface state information for peer leaf switches.
show mac address-table vlan *vlan-id*	Displays MAC addresses for a VLAN.
show platform software fed switch active matm macTable vlan *vlan-id*	Displays MAC addresses for a VLAN from MAC address table manager database for Forwarding Engine Driver (FED).
show device-tracking database	Displays device tracking database.
show device-tracking database mac	Displays device tracking MAC address database.
show ip mroute	Displays multicast routing table information.

CHAPTER 3

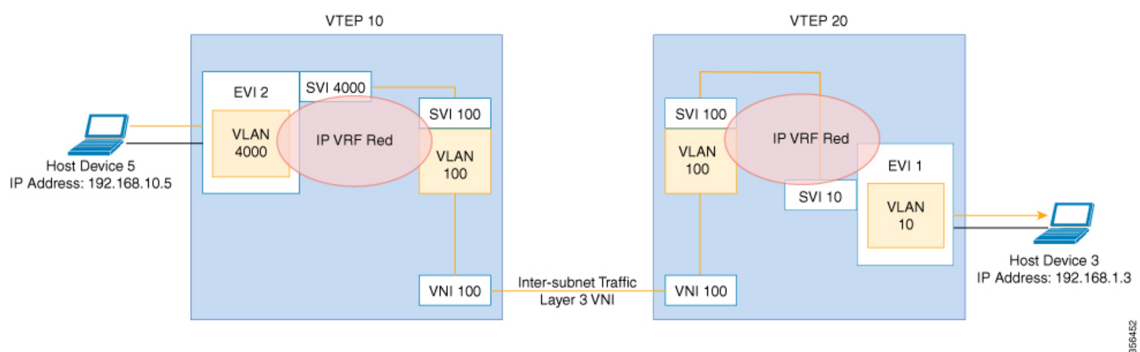
Configuring EVPN VXLAN Layer 3 Overlay Network

Information About EVPN VXLAN Layer 3 Overlay Network

An EVPN VXLAN Layer 3 overlay network allows host devices in different Layer 2 networks to send Layer 3 or routed traffic to each other. The network forwards the routed traffic using a Layer 3 virtual network instance (VNI) and an IP VRF.

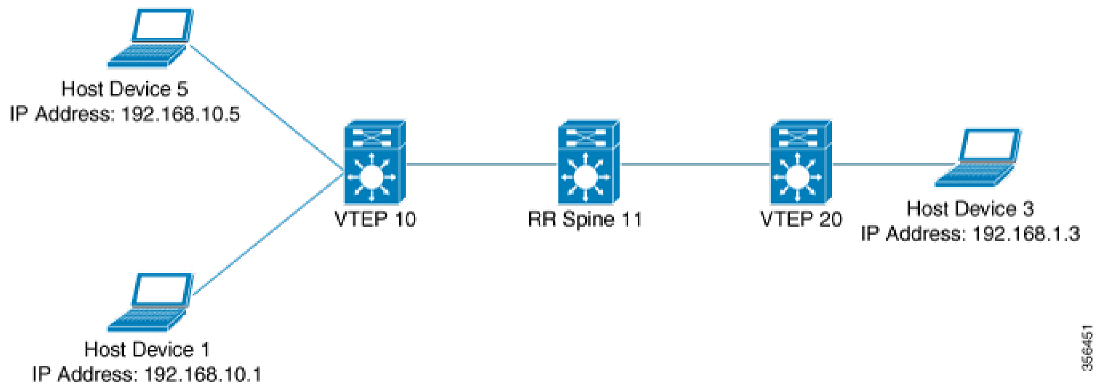
This module provides information only about how to configure a Layer 3 overlay network. You can also configure both Layer 2 and Layer 3 overlay networks together to enable integrated routing and bridging (IRB). For more information about IRB, see *Configuring EVPN VXLAN Integrated Routing and Bridging* module.

The following figure shows the movement of traffic in an EVPN VXLAN Layer 3 overlay network using a Layer 3 VNI:



How to Configure EVPN VXLAN Layer 3 Overlay Network

The following figure shows a sample topology of an EVPN VXLAN Network. Host device 3 and host device 5 are part of different subnets. The network forwards traffic from host device 1 to host device 3 using a Layer 3 VNI and an IP VRF.



Perform the following set of procedures to configure an EVPN VXLAN Layer 3 overlay network:

- Configure the IP VRF on the VTEPs.
- Configure the core-facing VLAN on the VTEPs.
- Configure the access-facing VLAN on the VTEPs.
- Configure the switch virtual interface (SVI) for the core-facing VLAN.
- Configure the SVI for the access-facing VLAN.
- Configure the loopback interface on the VTEPs.
- Configure the network virtualization endpoint (NVE) interface on the VTEPs.
- Configure BGP with either IPv4 or IPv6 or both address families on the VTEPs.

Configuring an IP VRF on a VTEP

To configure an IP VRF on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**vrf definition** *vrf-name* **Example:** `Device(config)# vrf definition Green`	Enters the VRF configuration mode for the specified VRF instance.
Step 4	**rd** *vpn-route-distinguisher* **Example:** `Device(config-vrf)# rd 100:1`	Specifies the route distinguisher for the VRF instance.
Step 5	**address-family ipv4** [**multicast** | **unicast**] **Example:** `Device(config-vrf)# address-family ipv4`	Enters the IPv4 address family configuration mode.

Step 6	**route-target** {**export** | **import** | **both**} *route-target-ext-community*
	Example:
	`Device(config-vrf-af)# route-target export 100:1`
	Example:
	`Device(config-vrf-af)# route-target import 100:1`
		Creates a list of import, export, or both import and export route target communities for the specified VRF.
		Enter either an autonomous system number and an arbitrary number (xxx:y), or an IP address and an arbitrary number (A.B.C.D:y).

Step 7	**route-target** {**export** | **import** | **both**} *route-target-ext-community* **stitching**
	Example:
	`Device(config-vrf-af)# route-target export 100:1 stitching`
	Example:
	`Device(config-vrf-af)# route-target import 100:1 stitching`
		Configures importing, exporting, or both importing and exporting of EVPN route target communities for the VRF.

Step 8	**exit-address-family** **Example:** `Device(config-vrf-af)# exit-address-family`	Exits VRF address family configuration mode and enters VRF configuration mode.
Step 9	**address-family ipv6** [**multicast** | **unicast**] **Example:** `Device(config-vrf)# address-family ipv6`	Enters the IPv6 address family configuration mode.
Step 10	**route-target** {**export** | **import** | **both**} *route-target-ext-community* **Example:**	Creates a list of import, export, or both import and export route target communities for the specified VRF.

	Command or Action	Purpose
	Device(config-vrf-af)# **route-target export 100:1** **Example:** Device(config-vrf-af)# **route-target import 100:1**	Enter either an autonomous system number and an arbitrary number (xxx:y), or an IP address and an arbitrary number (A.B.C.D:y).
Step 11	**route-target** {**export** | **import** | **both**} *route-target-ext-community* **stitching** **Example:** Device(config-vrf-af)# **route-target export 100:1 stitching** **Example:** Device(config-vrf-af)# **route-target import 100:1 stitching**	Configures importing, exporting, or both importing and exporting of VXLAN route target communities for the VRF.
Step 12	**exit-address-family** **Example:** Device(config-vrf-af)# **exit-address-family**	Exits VRF address family configuration mode and enters VRF configuration mode.
Step 13	**end** **Example:** Device(config-vrf)# **end**	Returns to privileged EXEC mode.

Configuring the Core-facing VLAN on a VTEP

To configure the core-facing VLAN on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** Device> **enable**	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** Device# **configure terminal**	Enters global configuration mode.
Step 3	**vlan configuration** *vlan-id* **Example:** Device(config)# **vlan configuration 11**	Enters VLAN feature configuration mode for the specified VLAN interface.

	Command or Action	Purpose
Step 4	**member vni** *l3-vni-number* **Example:** `Device(config-vlan)# member vni 5000`	Adds EVPN instance as a member of the VLAN configuration. The VNI here is used as a Layer 3 VNI.
Step 5	**end** **Example:** `Device(config-vlan)# end`	Returns to privileged EXEC mode.

Configuring Access-facing VLAN on a VTEP

To configure the access-facing VLAN on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-name* **Example:** `Device(config)# interface GigabitEthernet1/0/1`	Enters interface configuration mode for the specified interface.
Step 4	**switchport access vlan** *vlan-id* **Example:** `Device(config-if)# switchport access vlan 40`	Configures the interface as a static-access port of the specified VLAN. Interface can also be configured as a trunk interface, if required.
Step 5	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring Switch Virtual Interface for the Core-facing VLAN

To configure an SVI for the core-facing VLAN on the VTEP:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface vlan** *vlan-id* **Example:** `Device(config)# interface vlan 11`	Enters interface configuration mode for the specified VLAN.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding Green`	Configures the SVI for the VLAN.
Step 5	**ip unnumbered***Loopback-interface* **Example:** `Device(config-if)# ip unnumbered Loopback0`	Enables IP processing on the Loopback interface without assigning an explicit IP address to the interface.
Step 6	**no autostate** **Example:** `Device(config-if)# no autostate`	Disables autostate on the interface. In EVPN deployments, once a VLAN is used for a core-facing SVI, it should not be allowed in any trunk. For a core-facing SVI to function properly, the **no autostate** command must be configured under the SVI.
Step 7	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring the Switch Virtual Interface for the Access-facing VLANs

To configure the SVI for the access-facing VLAN on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface vlan** *vlan-id* **Example:** `Device(config)# interface vlan 40`	Enters interface configuration mode for the specified VLAN.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding Green`	Configures the SVI for the VLAN.
Step 5	**ip address***ip-address* **Example:** `Device(config-if)# ip address 192.168.10.100 255.255.255.0`	Configures the IP address of the SVI.
Step 6	**mac-address***mac-address-value* **Example:** `Device(config-if)# mac-address aabb.cc01.f100`	(Optional) Manually sets the MAC address for the VLAN interface.
Step 7	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.
Step 8	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring the Loopback Interface on a VTEP

To configure the loopback interface on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *loopback-interface-id* **Example:** `Device(config)# interface Loopback0`	Enters interface configuration mode for the specified Loopback interface.
Step 4	**ip address** *ipv4-address* **Example:** `Device(config-if)# ip address 10.12.11.11 255.255.255.255`	Configures the IP address for the Loopback interface.
Step 5	**ip pim sparse mode** **Example:** `Device(config-if)# ip pim sparse mode`	(Optional) Enables Protocol Independent Multicast (PIM) sparse mode on the Loopback interface. **Note** Enable PIM sparse mode only if EVPN VXLAN Layer 2 overlay network is also configured on the VTEP with underlay multicast as the mechanism for forwarding BUM traffic.
Step 6	**end** **Example:** `Device(config-vlan)# end`	Returns to privileged EXEC mode.

Configuring the NVE Interface on a VTEP

To add a Layer 3 VNI member to the NVE interface on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *nve-interface-id* **Example:** `Device(config)# interface nve1`	Defines the interface to be configured as a trunk, and enters interface configuration mode.
Step 4	**no ip address** **Example:**	Disables IP processing on the interface by removing its IP address.

	Command or Action	Purpose
	`Device(config-if)# no ip address`	
Step 5	**source-interface** *loopback-interface-id* **Example:** `Device(config-if)# source-interface loopback0`	Sets the IP address of the specified loopback interface as the source IP address.
Step 6	**host-reachability protocol bgp** **Example:** `Device(config-if)# host-reachability protocol bgp`	Configures BGP as the host-reachability protocol on the interface.
Step 7	**member vni** *vni-id* **vrf** *vrf-name* **Example:** `Device(config-if)# member vni 5000 vrf Green`	Associates the Layer 3 VNI id with the NVE interface. **Note** The Layer 3 VNI id must match with the VNI id configured in the core VLAN on the VTEP.
Step 8	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring BGP with IPv4 or IPv6 or Both Address Families on VTEP

To configure BGP on a VTEP with IPv4 or IPv6 or both address families and a spine switch as the neighbor, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 1`	Enables a BGP routing process, assigns it an autonomous system number, and enters router configuration mode.

	Command or Action	Purpose
Step 4	**bgp log-neighbor-changes** **Example:** `Device(config-router)# bgp log-neighbor-changes`	(Optional) Enables the generation of logging messages when the status of a BGP neighbor changes. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.
Step 5	**bgp update-delay** *time-period* **Example:** `Device(config-router)# bgp update-delay 1`	(Optional) Sets the maximum initial delay period before sending the first update. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.
Step 6	**bgp graceful-restart** **Example:** `Device(config-router)# bgp graceful-restart`	(Optional) Enables the BGP graceful restart capability for all BGP neighbors. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.
Step 7	**no bgp default ipv4-unicast** **Example:** `Device(config-router)# no bgp default ipv4-unicast`	(Optional) Disables default IPv4 unicast address family for BGP peering session establishment. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.
Step 8	**neighbor** *ip-address* **remote-as** *number* **Example:** `Device(config-router)# neighbor 10.11.11.11 remote-as 1`	Defines multiprotocol-BGP neighbors. Under each neighbor, define the configuration. Use the IP address of the spine switch as the neighbor IP address.
Step 9	**neighbor** {*ip-address* | *group-name*} **update-source** *interface* **Example:** `Device(config-router)# neighbor 10.11.11.11 update-source Loopback0`	Configures update source. Update source can be configured per neighbor or per peer-group. Use the IP address of the spine switch as the neighbor IP address.
Step 10	**address-family l2vpn evpn** **Example:** `Device(config-router)# address-family l2vpn evpn`	Specifies the L2VPN address family and enters address family configuration mode.
Step 11	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 10.11.11.11 activate`	Enables the exchange information from a BGP neighbor. Use the IP address of the spine switch as the neighbor IP address.
Step 12	**neighbor** *ip-address* **send-community** [**both** | **extended** | **standard**] **Example:**	Specifies the communities attribute sent to a BGP neighbor. Use the IP address of the spine switch as the neighbor IP address.

	Command or Action	Purpose
	`Device(config-router-af)# neighbor 10.11.11.11 send-community both`	
Step 13	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 14	**address-family ipv4 vrf** *vrf-name* **Example:** `Device(config-router)# address-family ipv4 vrf Green`	Specifies the IPv4 address family and enters address family configuration mode.
Step 15	**advertise l2vpn evpn** **Example:** `Device(config-router-af)# advertise l2vpn evpn`	Advertises Layer 2 VPN EVPN routes within a tenant VRF in an EVPN VXLAN fabric.
Step 16	**redistribute connected** **Example:** `Device(config-router-af)# redistribute connected`	(Optional) Redistributes connected routes to BGP.
Step 17	**redistribute static** **Example:** `Device(config-router-af)# redistribute static`	(Optional) Redistributes static routes to BGP.
Step 18	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 19	**address-family ipv6 vrf** *vrf-name* **Example:** `Device(config-router)# address-family ipv6 vrf green`	Specifies the IPv6 address family and enters address family configuration mode.
Step 20	**advertise l2vpn evpn** **Example:** `Device(config-router-af)# advertise l2vpn evpn`	Advertises Layer 2 VPN EVPN routes within a tenant VRF in an EVPN VXLAN fabric.
Step 21	**redistribute connected** **Example:** `Device(config-router-af)# redistribute connected`	(Optional) Redistributes connected routes to BGP.

	Command or Action	Purpose
Step 22	**redistribute static** **Example:** `Device(config-router-af)# redistribute static`	(Optional) Redistributes static routes to BGP.
Step 23	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 24	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.

Configuration Examples for EVPN VXLAN Layer 3 Overlay Network

This section provides an example for configuring an EVPN VXLAN Layer 3 overlay network. This example shows a sample configuration for a VXLAN network with 2 VTEPs, VTEP 1 and VTEP 2, connected to perform routing.

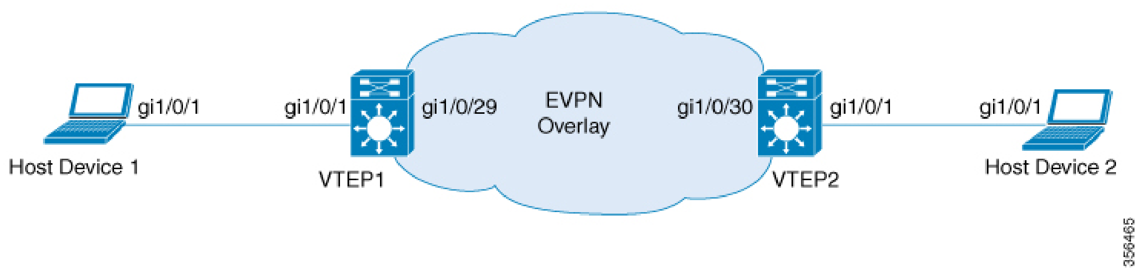


Table 3: Configuration Example for a VXLAN Network with Two VTEPs Connected to Perform Routing

VTEP 1	VTEP 2

VTEP 1	VTEP 2

VTEP1# **show running-config**
!
hostname VTEP1
!
!
vrf definition green
 rd 103:2
 !
 address-family ipv4
  route-target export 103:2
  route-target import 104:2
  route-target export 103:2 stitching
  route-target import 104:2 stitching
 exit-address-family
 !
 address-family ipv6
  route-target export 103:2
  route-target import 104:2
  route-target export 103:2 stitching
  route-target import 104:2 stitching
 exit-address-family
!
ip multicast-routing
ipv6 unicast-routing
!
!
system mtu 9150
!
vlan configuration 200
 member vni 5000
!
!
interface Loopback0
 ip address 10.1.1.10 255.255.255.255
 ip pim sparse-mode
!
interface Loopback13
 description demo only (for rt5 distribution)
 vrf forwarding green
 ip address 10.1.13.13 255.255.255.0
!
interface GigabitEthernet1/0/1
description access interface
 switchport access vlan 201
 switchport mode access
!
!
interface GigabitEthernet1/0/29
 description core-underlay-interface
 no switchport
 ip address 172.16.1.29 255.255.255.0
 ip pim sparse-mode
!
!
interface Vlan200
 description core svi for l3vni
 vrf forwarding green
 ip unnumbered Loopback0
 ipv6 enable
 no autostate
!
interface Vlan201
	VTEP2# **show running-config**
	!
	hostname VTEP2
	!
	!
	vrf definition green
	 rd 104:2
	 !
	 address-family ipv4
	  route-target export 104:2
	  route-target import 103:2
	  route-target export 104:2 stitching
	  route-target import 103:2 stitching
	 exit-address-family
	 !
	 address-family ipv6
	  route-target export 104:2
	  route-target import 103:2
	  route-target export 104:2 stitching
	  route-target import 103:2 stitching
	 exit-address-family
	!
	ip multicast-routing
	ipv6 unicast-routing
	!
	!
	system mtu 9150
	!
	vlan configuration 200
	 member vni 5000
	!
	!
	interface Loopback0
	 ip address 10.2.2.20 255.255.255.255
	 ip pim sparse-mode
	!
	interface Loopback14
	 description demo only (for rt5 distribution)
	 vrf forwarding green
	 ip address 10.1.14.14 255.255.255.0
	!
	interface GigabitEthernet1/0/1
	description access interface
	 switchport access vlan 202
	 switchport mode access
	!
	!
	interface GigabitEthernet1/0/30
	 description core-underlay-interface
	 no switchport
	 ip address 172.16.1.30 255.255.255.0
	 ip pim sparse-mode
	!
	!
	interface Vlan200
	 description core svi for l3vni
	 vrf forwarding green
	 ip unnumbered Loopback0
	 ipv6 enable
	 no autostate
	!
	interface Vlan202

<table>
<tr><th>VTEP 1</th><th>VTEP 2</th></tr>
<tr><td>

```
 description access-svi
 vrf forwarding green
 ip address 192.168.1.201 255.255.255.0
 ipv6 address 2001:DB8:201::201/64
 ipv6 enable
!
interface nve10
 no ip address
 source-interface Loopback0
 host-reachability protocol bgp
 member vni 5000 vrf green
!
router ospf 1
 router-id 10.1.1.10
 network 10.1.1.0 0.0.0.255 area 0
 network 172.16.1.0 0.0.0.255 area 0
!
router bgp 10
 bgp router-id interface Loopback0
 bgp log-neighbor-changes
 bgp update-delay 1
 no bgp default ipv4-unicast
 neighbor 10.2.2.20 remote-as 10
 neighbor 10.2.2.20 update-source Loopback0
 !
 address-family ipv4
 exit-address-family
 !
 address-family l2vpn evpn
  neighbor 10.2.2.20 activate
  neighbor 10.2.2.20 send-community both
 exit-address-family
 !
 address-family ipv4 vrf green
  advertise l2vpn evpn
  redistribute connected
  redistribute static
 exit-address-family
 !
 address-family ipv6 vrf green
  redistribute connected
  redistribute static
  advertise l2vpn evpn
 exit-address-family
!
ip pim rp-address 10.1.1.10
!
!
end
```

</td><td>

```
 description access-svi
 vrf forwarding green
 ip address 192.168.2.202 255.255.255.0
 ipv6 address 2001:DB8:202::202/64
 ipv6 enable
!
interface nve10
 no ip address
 source-interface Loopback0
 host-reachability protocol bgp
 member vni 5000 vrf green
!
router ospf 1
 router-id 10.2.2.20
 network 10.2.2.0 0.0.0.255 area 0
 network 172.16.1.0 0.0.0.255 area 0
!
router bgp 10
 bgp router-id interface Loopback0
 bgp log-neighbor-changes
 bgp update-delay 1
 no bgp default ipv4-unicast
 neighbor 10.1.1.10 remote-as 10
 neighbor 10.1.1.10 update-source Loopback0
 !
 address-family ipv4
 exit-address-family
 !
 address-family l2vpn evpn
  neighbor 10.1.1.10 activate
  neighbor 10.1.1.10 send-community both
 exit-address-family
 !
 address-family ipv4 vrf green
  advertise l2vpn evpn
  redistribute connected
  redistribute static
 exit-address-family
 !
 address-family ipv6 vrf green
  redistribute connected
  redistribute static
  advertise l2vpn evpn
 exit-address-family
!
ip pim rp-address 10.1.1.10
!
!
end
```

</td></tr>
</table>

The following examples provide outputs for **show** commands on VTEP 1 and VTEP 2 in the topology configured above.

- show nve peers, on page 46
- show bgp l2vpn evpn all, on page 46
- show ip route vrf, on page 47
- show platform software fed switch active matm mactable vlan, on page 48

show nve peers

VTEP 1

The following example shows the output for the **show nve peers** command on VTEP 1:

```
VTEP1# show nve peers
Interface  VNI      Type Peer-IP          RMAC/Num_RTs   eVNI     state flags UP time
nve10      5000     L3CP 10.2.2.20        380e.4d9b.6a4a 5000       UP  A/M/4 00:38:37
nve10      5000     L3CP 10.2.2.20        380e.4d9b.6a4a 5000       UP  A/-/6 00:03:16
```

VTEP 2

The following example shows the output for the **show nve peers** command on VTEP 2:

```
VTEP2# show nve peers
Interface  VNI      Type Peer-IP          RMAC/Num_RTs   eVNI     state flags UP time
nve10      5000     L3CP 10.1.1.10        a0f8.4910.bce2 5000       UP  A/-/4 00:38:53
nve10      5000     L3CP 10.1.1.10        a0f8.4910.bce2 5000       UP  A/M/6 00:38:53
```

show bgp l2vpn evpn all

VTEP 1

The following example shows the output for the **show bgp l2vpn evpn all** all command on VTEP 1:

```
VTEP1# show bgp l2vpn evpn all
BGP table version is 26, local router ID is 10.1.1.10
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path, L long-lived-stale,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 103:2 (default for vrf green)
 *>   [5][103:2][0][24][10.1.13.0]/17
                      0.0.0.0                  0         32768 ?
 *>   [5][103:2][0][24][192.168.1.0]/17
                      0.0.0.0                  0         32768 ?
 *>   [5][103:2][0][64][2001:DB8:201::]/29
                      ::                       0         32768 ?
Route Distinguisher: 104:2
 *>i  [5][104:2][0][24][10.1.14.0]/17
                      10.2.2.20                0    100      0 ?
 *>i  [5][104:2][0][24][192.168.2.0]/17
                      10.2.2.20                0    100      0 ?
 *>i  [5][104:2][0][64][2001:DB8:202::]/29
                      10.2.2.20                0    100      0 ?
```

VTEP 2

The following example shows the output for the **show bgp l2vpn evpn all** command on VTEP 2:

```
VTEP2# show bgp l2vpn evpn all
BGP table version is 12, local router ID is 10.2.2.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path, L long-lived-stale,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 103:2
 *>i  [5][103:2][0][24][10.1.13.0]/17
                      10.1.1.10                0    100      0 ?
 *>i  [5][103:2][0][24][192.168.1.0]/17
                      10.1.1.10                0    100      0 ?
 *>i  [5][103:2][0][64][2001:DB8:201::]/29
                      10.1.1.10                0    100      0 ?
Route Distinguisher: 104:2 (default for vrf green)
 *>   [5][104:2][0][24][10.1.14.0]/17
                      0.0.0.0                  0         32768 ?
 *>   [5][104:2][0][24][192.168.2.0]/17
                      0.0.0.0                  0         32768 ?
 *>   [5][104:2][0][64][2001:DB8:202::]/29
     Network          Next Hop            Metric LocPrf Weight Path
                      ::                       0         32768 ?
```

show ip route vrf

VTEP 1

The following example shows the output for the **show ip route vrf** command on VTEP 1:

```
VTEP1# show ip route vrf green
Routing Table: green
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C        10.1.13.0/24 is directly connected, Loopback13
L        10.1.13.13/32 is directly connected, Loopback13
B        10.1.14.0/24 [200/0] via 10.2.2.20, 00:42:01, Vlan200
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.1.0/24 is directly connected, Vlan201
L        192.168.1.201/32 is directly connected, Vlan201
B     192.168.2.0/24 [200/0] via 10.2.2.20, 00:06:00, Vlan200
```

VTEP 2

The following example shows the output for the **show ip route vrf** command on VTEP 2:

```
VTEP2# show ip route vrf green
Routing Table: green
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
B        10.1.13.0/24 [200/0] via 10.1.1.10, 00:42:38, Vlan200
C        10.1.14.0/24 is directly connected, Loopback14
L        10.1.14.14/32 is directly connected, Loopback14
B     192.168.1.0/24 [200/0] via 10.1.1.10, 00:42:38, Vlan200
      192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.2.0/24 is directly connected, Vlan202
L        192.168.2.202/32 is directly connected, Vlan202
```

show platform software fed switch active matm mactable vlan

VTEP 1

The following example shows the output for the **show platform software fed switch active matm mactable vlan 200** command on VTEP 1:

Note The MAC address of the peer's core SVI interface must be present in the core VLAN.

```
VTEP1# show platform software fed switch active matm macTable vlan 200
VLAN   MAC                  Type  Seq#   EC_Bi  Flags  machandle          siHandle
      riHandle            diHandle              *a_time  *e_time  ports
------------------------------------------------------------------------------------------
200    a0f8.4910.bce2      0x8002      0  19880     64  0x7f5d8503fd48      0x7f5d852b6d28
      0x0                0x5234                     0        0  Vlan200

200    380e.4d9b.6a4a   0x1000001      0      0     64  0x7f5d85117598      0x7f5d85110f78
      0x7f5d851b9648      0x0                       0        0  RLOC 10.2.2.20 adj_id 22


Total Mac number of addresses:: 2
```

VTEP 2

The following example shows the output for the **show platform software fed switch active matm mactable vlan 200** command on VTEP 2:

Note The MAC address of the peer's core SVI interface must be present in the core VLAN.

```
VTEP2# show platform software fed switch active matm macTable vlan 200
VLAN   MAC                     Type  Seq#   EC_Bi  Flags  machandle           siHandle
      riHandle            diHandle              *a_time  *e_time  ports

------------------------------------------------------------------------------------------
200    380e.4d9b.6a4a      0x8002      0  42949     64  0x7f40e15fd308      0x7f40e15f49d8
      0x0                 0x0                          0        0  Vlan200

200    a0f8.4910.bce2   0x1000001      0      0     64  0x7f40e193c478      0x7f40e1938168
      0x7f40e1937bf8      0x0                          0        0  RLOC 10.1.1.10 adj_id 86

Total Mac number of addresses:: 2
```

Verifying EVPN VXLAN Layer 3 Overlay Network

The following table lists the **show** commands that are used to verify a Layer 3 VXLAN overlay network:

Table 4: Commands to Verify EVPN VXLAN Layer 3 Overlay Network

Command	Purpose
show nve vni	Displays information about VXLAN network identifier members associated with an NVE interface.
show nve vni *vni-id* **detail**	Displays detailed NVE interface state information for a VXLAN network identifier member.
show nve peers	Displays NVE interface state information for peer leaf switches.
show mac address-table vlan *vlan-id*	Displays MAC addresses for a VLAN.
show platform software fed switch active matm macTable vlan *vlan-id*	Displays MAC addresses for a VLAN from MAC address table manager database for Forwarding Engine Driver (FED).
show ip route vrf *vrf-name*	Displays the IP routing table associated with a specific VRF.
show ip cef vrf *vrf-name*	Displays entries in the Cisco Express Forwarding (CEF) table associated with a VRF.
show arp vrf *vrf-name*	Displays entries in the Address Resolution Protocol (ARP) table associated with a VRF.
show bgp l2vpn evpn route-type 5	Displays BGP information for route type 5 of Layer 2 VPN EVPN address family.

Command	Purpose
show bgp l2vpn evpn all	Displays all BGP information for L2VPN EVPN address family.

CHAPTER 4

Configuring EVPN VXLAN Integrated Routing and Bridging

Information About EVPN VXLAN Integrated Routing and Bridging

EVPN VXLAN integrated routing and bridging (IRB) allows the VTEPs or leaf switches in an EVPN VXLAN network to perform both bridging and routing. IRB allows the VTEPs to forward both Layer 2 or bridged and Layer 3 or routed traffic. A VTEP performs bridging when it forwards traffic to the same subnet. Similarly, a VTEP performs routing when it forwards traffic to a different subnet. The VTEPs in the network forward traffic to each other through the VXLAN gateways. BGP EVPN VXLAN implements IRB in two ways:

- Asymmetric IRB
- Symmetric IRB

Asymmetric IRB

In asymmetric IRB, the ingress VTEP performs both bridging and routing whereas the egress VTEP performs only bridging. A packet first moves through a MAC VRF followed by an IP VRF on the network virtualisation endpoint (NVE) of the ingress VTEP. It then moves only through a MAC VRF on the NVE of the egress VTEP. The NVE of the ingress VTEP manages all the packet processing associated with intersubnet forwarding semantics.

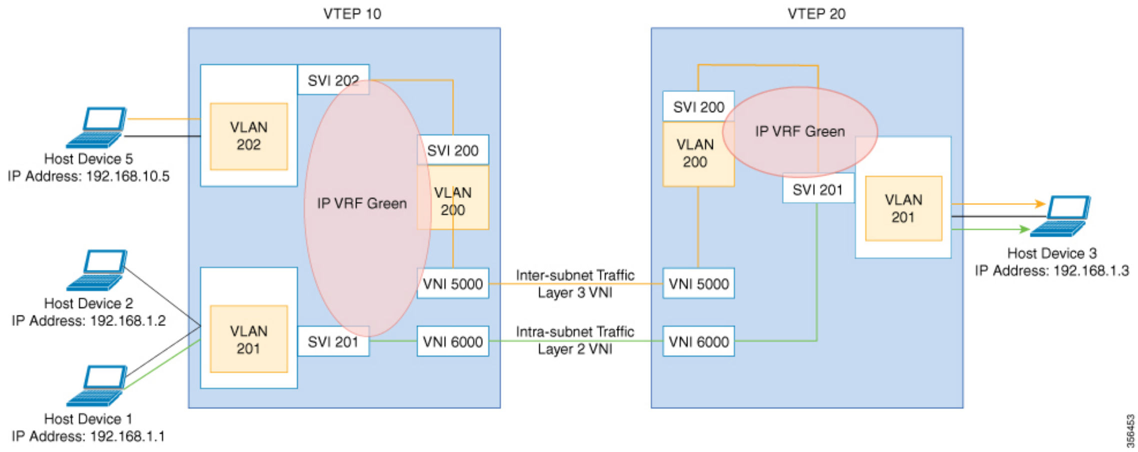
The return traffic during asymmetric IRB goes through a different virtual network instance (VNI) compared to the source traffic. Asymmetric IRB needs the source and destination VNIs to be associated with both the ingress and egress VTEPs.

Symmetric IRB

In symmetric IRB, both the ingress and egress VTEPs perform both bridging and routing. A packet first moves through a MAC VRF followed by an IP VRF on the NVE of the ingress VTEP. It then moves through an IP VRF followed by a MAC VRF on the NVE of the egress VTEP. The NVEs of ingress and egress VTEPs equally share all the packet processing associated with intersubnet forwarding semantics.

In symmetric IRB, you are required to define only the VNIs of locally attached endpoints on the ingress and egress VTEPs. Symmetric IRB offers better scalability in terms of the number of VNIs that a BGP EVPN VXLAN fabric supports.

The following figure shows the implementation of symmetric IRB and the movement of traffic in an EVPN VXLAN network:

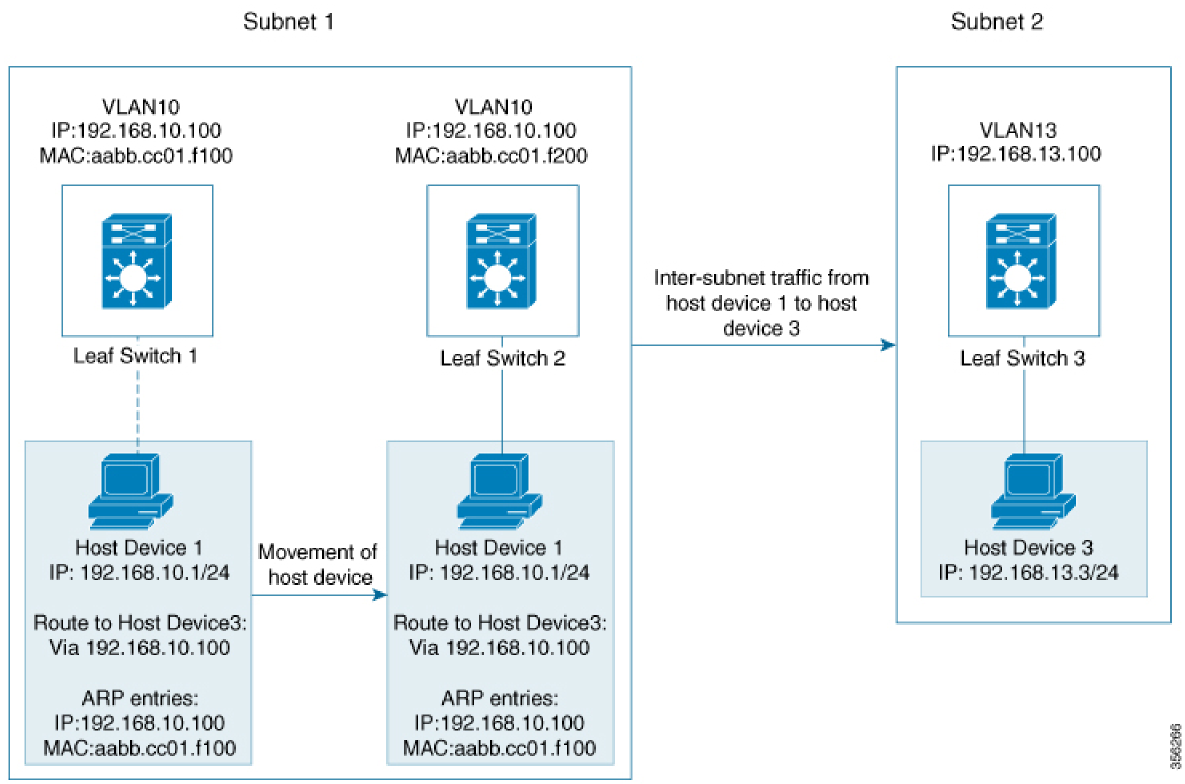


EVPN VXLAN Distributed Anycast Gateway

Distributed anycast gateway is a default gateway addressing mechanism in a BGP EVPN VXLAN fabric. The feature enables the use of the same gateway IP and MAC address across all the VETPs in an EVPN VXLAN network. This ensures that every VTEP functions as the default gateway for the workloads directly connected to it. The feature facilitates flexible workload placement, host mobility, and optimal traffic forwarding across the BGP EVPN VXLAN fabric.

The scenario shown in the following figure depicts a distributed gateway. Subnet 1 contains two leaf switches, leaf switch 1 and leaf switch 2, acting together as a distributed default gateway for VLAN 10. Host device 1 is connected to leaf switch 1 and needs to send traffic to host device 3, which is in a different subnet. When host device 1 tries to send traffic outside of subnet 1, the traffic goes through the configured gateway on leaf switch 1. Host device 1 registers the Address Resolution Protocol (ARP) entries of the gateway VLAN MAC and IP addresses on leaf switch 1.

Figure 1: Distributed Gateway Topology



When multiple VETPs act together as one single distributed default gateway for the same VLAN, the VLAN IP address remains the same across all of them. This IP address becomes the gateway IP address for any host device in the VLAN that tries to reach an IP address outside its subnet. But, each VTEP retains its own MAC address.

In the preceding figure, consider the scenario where host device 1 moves from leaf switch 1 to leaf switch 2. The host device remains within the same network and still maintains the same ARP entries for gateway MAC and IP addresses. But the MAC addresses of the VLAN interfaces on leaf switch 2 and leaf switch 1 are different. This results in a MAC address mismatch between the ARP entry and the VLAN on leaf switch 2. As a result, any traffic that host device 1 tries to send outside of Subnet 1 is either lost or continuously flooded as unknown unicast. EVPN VXLAN distributed anycast gateway feature prevents this traffic loss by ensuring that all the VTEPs have the same gateway MAC and IP addresses.

There are two ways to maintain the same MAC address across all VTEPs and configure distributed anycast gateway:

- Manual MAC address configuration
- MAC aliasing

Manual MAC Address Configuration

Manual MAC address configuration is the conventional method of enabling distributed anycast gateway in an EVPN VXLAN network. In this method, you manually configure the same MAC address on the Layer 2

VNI VLAN SVI on all the VTEPs in the network. You must configure the same MAC address on all the VTEPs in the same Layer 2 VNI.

Note The VLAN SVIs on all the leaf switches must already share the same gateway IP address.

In the Figure 1: Distributed Gateway Topology, on page 53 image, to enable distributed anycast gateway in subnet 1, configure the same MAC address on leaf switch 1 and leaf switch 2. This ensures that the ARP entries of gateway MAC and IP addresses on host device 1 match with the MAC and IP addresses of both leaf switch 1 and leaf switch 2.

MAC Aliasing

MAC aliasing for distributed anycast gateway removes the need to configure the same MAC address explicitly on the VLAN interfaces of every VTEP. MAC aliasing allows the VTEPs to advertise their VLAN MAC addresses as the gateway MAC addresses to all the other VTEPs in the network. The VTEPs in the network store the advertised MAC address as a gateway MAC address provided their VLAN IP address matches with the gateway IP address.

In the Figure 1: Distributed Gateway Topology, on page 53 image, consider the scenario where MAC aliasing is enabled in subnet 1. Leaf switch 1 and leaf switch 2 advertise their MAC addresses to each other as gateway MAC addresses. This allows leaf switch 2 to recognize the MAC address in the ARP entry of host device 1 as a gateway MAC address. It allows host device 1 to send traffic outside of subnet 1 even though its VLAN MAC address does not match with the ARP entry.

MAC aliasing in an EVPN VXLAN network is configured by enabling the default gateway advertisement on all the VTEPs.

How to Configure EVPN VXLAN Integrated Routing and Bridging

To configure EVPN VXLAN IRB, you need to configure EVPN VXLAN Layer 2 and Layer 3 overlay networks, and enable the gateways in the VXLAN network.

To enable IRB in a VXLAN network using distributed anycast gateway, perform the following set of procedures:

- Configure Layer 2 VPN EVPN on the VTEPs.

  Enable distributed anycast gateway for the VXLAN network when you configure Layer 2 VPN.

- Configure the core-facing and access-facing VLANs on the VTEPs.
- Configure switch virtual interface (SVI) for the core-facing VLAN on the VTEPs.
- Configure SVI for the access-facing VLAN on the VTEPs.
- Configure the IP VRF on the VTEPs.
- Configure the Loopback interface on the VTEPs.
- Configure the Network Virtualization Endpoint (NVE) interface on the VTEPs.
- Configure BGP with EVPN address family on the VTEPs.

Configuring Layer 2 VPN EVPN on a VTEP

See Configuring Layer 2 VPN EVPN on a VTEP, on page 14 for detailed steps.

Configuring IP VRF on VTEP

See Configuring an IP VRF on a VTEP, on page 32 for detailed steps.

Configuring Core-facing and Access-facing VLANs on a VTEP

To configure the core-facing and access-facing VLANs on a VTEP and enable IRB in the EVPN VXLAN network, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vlan configuration** *vlan-id* **Example:** `Device(config)# vlan configuration 201`	Enters VLAN feature configuration mode for the specified VLAN interface.
Step 4	**member evpn-instance** *evpn-instance-id* **vni** *l2-vni-number* **Example:** `Device(config-vlan)# member evpn-instance 1 vni 6000`	Adds EVPN instance as a member of the VLAN configuration. The VNI here is used as a Layer 2 VNI.
Step 5	**exit** **Example:** `Device(config-vlan)# exit`	Returns to global configuration mode.
Step 6	**vlan configuration** *vlan-id* **Example:** `Device(config)# vlan configuration 202`	Enters VLAN feature configuration mode for the specified VLAN interface.
Step 7	**member evpn-instance** *evpn-instance-id* **vni** *l2-vni-number* **Example:**	Adds EVPN instance as a member of the VLAN configuration. The VNI here is used as a Layer 2 VNI.

	Command or Action	Purpose
	`Device(config-vlan)# member evpn-instance 2 vni 7000`	
Step 8	**exit** **Example:** `Device(config-vlan)# exit`	Returns to global configuration mode.
Step 9	**vlan configuration** *vlan-id* **Example:** `Device(config)# vlan configuration 200`	Enters VLAN feature configuration mode for the specified VLAN interface.
Step 10	**member vni** *l3-vni-number* **Example:** `Device(config-vlan)# member vni 5000`	Adds EVPN instance as a member of the VLAN configuration. The VNI here is used as a Layer 3 VNI.
Step 11	**exit** **Example:** `Device(config-vlan)# exit`	Returns to global configuration mode.
Step 12	**end** **Example:** `Device(config-vlan)# end`	Returns to privileged EXEC mode.

Configuring Switch Virtual Interface for the Core-facing VLAN on a VTEP

To configure an SVI for the core-facing VLAN on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface vlan** *vlan-id* **Example:** `Device(config)# interface vlan 200`	Enters interface configuration mode for the specified VLAN.
Step 4	**vrf forwarding** *vrf-name* **Example:**	Configures the SVI for the VLAN.

	Command or Action	Purpose
	`Device(config-if)# vrf forwarding Green`	
Step 5	**ip unnumbered***Loopback-interface* **Example:** `Device(config-if)# ip unnumbered Loopback0`	Enables IP processing on the Loopback interface without assigning an explicit IP address to the interface.
Step 6	**no autostate** **Example:** `Device(config-if)# no autostate`	Disables autostate on the interface. In EVPN deployments, once a VLAN is used for a core-facing SVI, it should not be allowed in any trunk. For a core-facing SVI to function properly, the **no autostate** command must be configured under the SVI.
Step 7	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring Switch Virtual Interface for the Access-facing VLANs on a VTEP

To configure SVIs for the access-facing VLANs on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface vlan** *vlan-id* **Example:** `Device(config)# interface vlan 202`	Enters interface configuration mode for the specified VLAN.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding Green`	Configures the SVI for the VLAN.
Step 5	**ip address** *gateway-ip-address* **Example:** `Device(config-if)# ip address 192.168.10.1 255.255.255.0`	Configures the gateway IP address for the access SVI. Configure the same gateway IP address for this SVI on all the other VTEPs.

	Command or Action	Purpose
Step 6	**mac-address** *mac-address-value* **Example:** `Device(config-if)# mac-address aabb.cc01.f100`	(Optional) Manually sets the MAC address for the VLAN interface. To configure distributed anycast gateway in a VXLAN network using manual MAC configuration, configure the same MAC address on the corresponding Layer 2 VNI SVIs on all the VTEPs in a VXLAN network.
Step 7	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring the Loopback Interface on a VTEP

See Configuring the Loopback Interface on a VTEP, on page 37 for detailed steps.

Configuring the NVE Interface on a VTEP

To add Layer 2 and Layer 3 VNI members to the NVE interface of a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *nve-interface-id* **Example:** `Device(config)# interface nve1`	Defines the interface to be configured as a trunk, and enters interface configuration mode.
Step 4	**no ip address** **Example:** `Device(config-if)# no ip address`	Disables IP processing on the interface by removing its IP address.
Step 5	**source-interface** *loopback-interface-id* **Example:** `Device(config-if)# source-interface loopback0`	Sets the IP address of the specified loopback interface as the source IP address.

	Command or Action	Purpose
Step 6	**host-reachability protocol bgp** **Example:** `Device(config-if)# host-reachability protocol bgp`	Configures BGP as the host-reachability protocol on the interface.

Step 7	**member vni** *layer2-vni-id* {**ingress-replication** | **mcast-group** *multicast-group-address*
	Example:
	`Device(config-if)# member vni 6000 mcast-group 227.0.0.1`
	`Device(config-if)# member vni 7000 mcast-group 227.0.0.1`
		Associates the Layer 2 VNI member with the NVE.
		The specified replication type must match the replication type that is configured globally or for the specific EVPN instance. Use **mcast-group** keyword for static replication and **ingress-replication** keyword for ingress replication.

Step 8	**member vni** *layer3-vni-id* **vrf** *vrf-name* **Example:** `Device(config-if)# member vni 5000 vrf Green`	Associates the Layer 3 VNI member with the NVE.
Step 9	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring BGP with EVPN and VRF Address Families on a VTEP

To configure BGP on a VTEP with EVPN and VRF address families and a spine switch as the neighbor, perform these steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 1`	Enables a BGP routing process, assigns it an autonomous system number, and enters router configuration mode.

	Command or Action	Purpose
Step 4	**bgp log-neighbor-changes** **Example:** `Device(config-router)# bgp log-neighbor-changes`	(Optional) Enables the generation of logging messages when the status of a BGP neighbor changes. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.
Step 5	**bgp update-delay** *time-period* **Example:** `Device(config-router)# bgp update-delay 1`	(Optional) Sets the maximum initial delay period before sending the first update. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.
Step 6	**bgp graceful-restart** **Example:** `Device(config-router)# bgp graceful-restart`	(Optional) Enables the BGP graceful restart capability for all BGP neighbors. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.
Step 7	**no bgp default ipv4-unicast** **Example:** `Device(config-router)# no bgp default ipv4-unicast`	(Optional) Disables default IPv4 unicast address family for BGP peering session establishment. For more information, see *Configuring BGP* module of the *IP Routing Configuration Guide*.
Step 8	**neighbor** *ip-address* **remote-as** *number* **Example:** `Device(config-router)# neighbor 10.11.11.11 remote-as 1`	Defines multiprotocol-BGP neighbors. Under each neighbor, define the Layer 2 Virtual Private Network (L2VPN) EVPN configuration. Use the IP address of the spine switch as the neighbor IP address.
Step 9	**neighbor** {*ip-address* | *group-name*} **update-source** *interface* **Example:** `Device(config-router)# neighbor 10.11.11.11 update-source Loopback0`	Configures update source. Update source can be configured per neighbor or per peer-group. Use the IP address of the spine switch as the neighbor IP address.
Step 10	**address-family l2vpn evpn** **Example:** `Device(config-router)# address-family l2vpn evpn`	Specifies the L2VPN address family and enters address family configuration mode.
Step 11	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 10.11.11.11 activate`	Enables the exchange information from a BGP neighbor. Use the IP address of the spine switch as the neighbor IP address.
Step 12	**neighbor** *ip-address* **send-community** [**both** | **extended** | **standard**]	Specifies the communities attribute sent to a BGP neighbor.

	Command or Action	Purpose
	Example: `Device(config-router-af)# neighbor 10.11.11.11 send-community both`	Use the IP address of the spine switch as the neighbor IP address.
Step 13	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 14	**address-family ipv4 vrf** *vrf-name* **Example:** `Device(config-router)# address-family ipv4 vrf green`	Specifies the IPv4 address family and enters address family configuration mode.
Step 15	**advertise l2vpn evpn** **Example:** `Device(config-router-af)# advertise l2vpn evpn`	Advertises Layer 2 VPN EVPN routes within a tenant VRF in an EVPN VXLAN fabric.
Step 16	**redistribute connected** **Example:** `Device(config-router-af)# redistribute connected`	Redistributes connected routes to BGP.
Step 17	**redistribute static** **Example:** `Device(config-router-af)# redistribute static`	Redistributes static routes to BGP.
Step 18	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 19	**address-family ipv6 vrf** *vrf-name* **Example:** `Device(config-router)# address-family ipv6 vrf green`	Specifies the IPv6 address family and enters address family configuration mode.
Step 20	**advertise l2vpn evpn** **Example:** `Device(config-router-af)# advertise l2vpn evpn`	Advertises Layer 2 VPN EVPN routes within a tenant VRF in an EVPN VXLAN fabric.
Step 21	**redistribute connected** **Example:**	Redistributes connected routes to BGP.

	Command or Action	Purpose
	`Device(config-router-af)# redistribute connected`	
Step 22	**redistribute static** **Example:** `Device(config-router-af)# redistribute static`	Redistributes static routes to BGP.
Step 23	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 24	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.

Configuration Examples for EVPN VXLAN Integrated Routing and Bridging

This section provides an example to show how to enable EVPN VXLAN IRB using distributed anycast gateway. The following example shows a sample configuration for a VXLAN network with 2 VTEPs. VTEP 1 and VTEP 2 are connected to perform integrated routing and bridging.

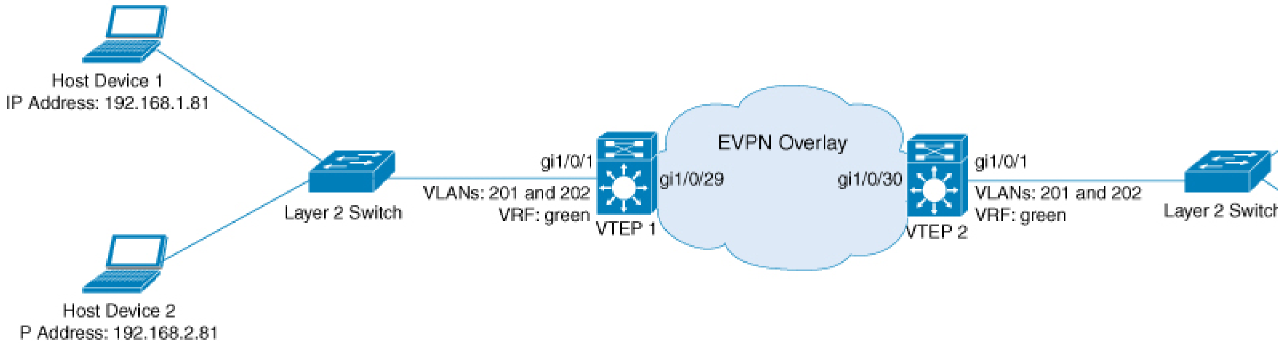


Table 5: Configuration Example for a VXLAN Network with Two VTEPs Connected to Perform Integrated Routing and Bridging Using Distributed Anycast Gateway

VTEP 1	VTEP 2

<table>
<tr><th>VTEP 1</th><th>VTEP 2</th></tr>
<tr><td><pre>
VTEP1# show running-config
!
hostname VTEP1
!
vrf definition green
 rd 103:2
 !
 address-family ipv4
  route-target export 103:2
  route-target import 104:2
  route-target export 103:2 stitching
  route-target import 104:2 stitching
 exit-address-family
 !
 address-family ipv6
  route-target export 103:2
  route-target import 104:2
  route-target export 103:2 stitching
  route-target import 104:2 stitching
 exit-address-family
!
ip routing
ip multicast-routing
ipv6 unicast-routing
!
!
l2vpn evpn
 replication-type static
 router-id Loopback0
 default-gateway advertise
!
l2vpn evpn instance 1 vlan-based
 encapsulation vxlan
!
l2vpn evpn instance 2 vlan-based
 encapsulation vxlan
!
!
system mtu 9150
!
vlan configuration 200
 member vni 5000
vlan configuration 201
 member evpn-instance 1 vni 6000
vlan configuration 202
 member evpn-instance 2 vni 7000
!
!
interface Loopback0
 ip address 10.1.1.10 255.255.255.255
 ip pim sparse-mode
!
interface Loopback13
description demo only (for rt5 distribution)
 vrf forwarding green
 ip address 10.1.13.13 255.255.255.0
!
interface GigabitEthernet1/0/1
description access-facing-interface
 switchport trunk allowed vlan 201,202
 switchport mode trunk
!
</pre></td><td><pre>
VTEP2# show running-config
!
hostname VTEP2
!
vrf definition green
 rd 104:2
 !
 address-family ipv4
  route-target export 104:2
  route-target import 103:2
  route-target export 104:2 stitching
  route-target import 103:2 stitching
 exit-address-family
 !
 address-family ipv6
  route-target export 104:2
  route-target import 103:2
  route-target export 104:2 stitching
  route-target import 103:2 stitching
 exit-address-family
!
ip routing
ip multicast-routing
ipv6 unicast-routing
!
!
l2vpn evpn
 replication-type static
 router-id Loopback0
 default-gateway advertise
!
l2vpn evpn instance 1 vlan-based
 encapsulation vxlan
!
l2vpn evpn instance 2 vlan-based
 encapsulation vxlan
!
!
system mtu 9150
!
vlan configuration 200
 member vni 5000
vlan configuration 201
 member evpn-instance 1 vni 6000
vlan configuration 202
 member evpn-instance 2 vni 7000
!
!
interface Loopback0
 ip address 10.2.2.20 255.255.255.255
 ip pim sparse-mode
!
interface Loopback14
 description demo only (for rt5 distribution)
 vrf forwarding green
 ip address 10.1.14.14 255.255.255.0
!
interface GigabitEthernet1/0/1
description access-facing-interface
 switchport trunk allowed vlan 201,202
 switchport mode trunk
!
</pre></td></tr>
</table>

VTEP 1	VTEP 2

<pre>!
interface GigabitEthernet1/0/29
 description core-underlay-interface
 no switchport
 ip address 172.16.1.29 255.255.255.0
 ip pim sparse-mode
!
!
interface Vlan200
 description core svi for l3vni
 vrf forwarding green
 ip unnumbered Loopback0
 ipv6 enable
 no autostate
!
interface Vlan201
 description vni 6000 default-gateway
 vrf forwarding green
 ip address 192.168.1.201 255.255.255.0
 ipv6 address 2001:DB8:201::201/64
 ipv6 enable
!
interface Vlan202
 description vni 7000 default-gateway
 vrf forwarding green
 ip address 192.168.2.202 255.255.255.0
 ipv6 address 2001:DB8:202::202/64
 ipv6 enable
!
!
interface nve10
 no ip address
 source-interface Loopback0
 host-reachability protocol bgp
 member vni 6000 mcast-group 232.1.1.1
 member vni 5000 vrf green
 member vni 7000 mcast-group 232.1.1.1
!
router ospf 1
 router-id 10.1.1.10
 network 10.1.1.0 0.0.0.255 area 0
 network 172.16.1.0 0.0.0.255 area 0
!
router bgp 10
 bgp router-id interface Loopback0
 bgp log-neighbor-changes
 bgp update-delay 1
 no bgp default ipv4-unicast
 neighbor 10.2.2.20 remote-as 10
 neighbor 10.2.2.20 update-source Loopback0
 !
 address-family ipv4
 exit-address-family
 !
 address-family l2vpn evpn
  neighbor 10.2.2.20 activate
  neighbor 10.2.2.20 send-community both
 exit-address-family
 !
 address-family ipv4 vrf green
  advertise l2vpn evpn
  redistribute connected</pre>
	<pre>!
	interface GigabitEthernet1/0/30
	 description core-underlay-interface
	 no switchport
	 ip address 172.16.1.30 255.255.255.0
	 ip pim sparse-mode
	!
	!
	interface Vlan200
	 description core svi for l3vni
	 vrf forwarding green
	 ip unnumbered Loopback0
	 ipv6 enable
	 no autostate
	!
	interface Vlan201
	 description vni 6000 default-gateway
	 vrf forwarding green
	 ip address 192.168.1.201 255.255.255.0
	 ipv6 address 2001:DB8:201::201/64
	 ipv6 enable
	!
	interface Vlan202
	 description vni 7000 default-gateway
	 vrf forwarding green
	 ip address 192.168.2.202 255.255.255.0
	 ipv6 address 2001:DB8:202::202/64
	 ipv6 enable
	!
	!
	interface nve10
	 no ip address
	 source-interface Loopback0
	 host-reachability protocol bgp
	 member vni 6000 mcast-group 232.1.1.1
	 member vni 7000 mcast-group 232.1.1.1
	 member vni 5000 vrf green
	!
	router ospf 1
	 router-id 10.2.2.20
	 network 10.2.2.0 0.0.0.255 area 0
	 network 172.16.1.0 0.0.0.255 area 0
	!
	router bgp 10
	 bgp router-id interface Loopback0
	 bgp log-neighbor-changes
	 bgp update-delay 1
	 no bgp default ipv4-unicast
	 neighbor 10.1.1.10 remote-as 10
	 neighbor 10.1.1.10 update-source Loopback0
	 !
	 address-family ipv4
	 exit-address-family
	 !
	 address-family l2vpn evpn
	  neighbor 10.1.1.10 activate
	  neighbor 10.1.1.10 send-community both
	 exit-address-family
	 !
	 address-family ipv4 vrf green
	  advertise l2vpn evpn
	  redistribute connected</pre>

VTEP 1	VTEP 2

`  redistribute static`
` exit-address-family`
` !`
` address-family ipv6 vrf green`
`  redistribute connected`
`  redistribute static`
`  advertise l2vpn evpn`
` exit-address-family`
`!`
`ip pim rp-address 10.1.1.10`
`!`
`end`
	`  redistribute static`
	` exit-address-family`
	` !`
	` address-family ipv6 vrf green`
	`  redistribute connected`
	`  redistribute static`
	`  advertise l2vpn evpn`
	` exit-address-family`
	`!`
	`ip pim rp-address 10.1.1.10`
	`!`
	`end`

The following examples provide outputs for **show** commands on VTEP 1 and VTEP 2 in the topology configured above:

- show nve peers, on page 66
- show l2vpn evpn peers vxlan, on page 67
- show l2vpn evpn evi evpn-instance detail, on page 67
- show l2vpn evpn default-gateway, on page 68
- show bgp l2vpn evpn all, on page 69
- show ip route vrf green, on page 72
- show platform software fed switch active matm mactable vlan, on page 73

show nve peers

VTEP 1

The following example shows the output for the **show nve peers** command on VTEP 1:

```
VTEP1# show nve peers
Interface  VNI      Type Peer-IP          RMAC/Num_RTs   eVNI     state flags UP time
nve10      5000     L3CP 10.2.2.20        380e.4d9b.6a4a 5000       UP  A/M/4 01:33:41
nve10      5000     L3CP 10.2.2.20        380e.4d9b.6a4a 5000       UP  A/-/6 00:43:38
nve10      6000     L2CP 10.2.2.20        5              6000       UP   N/A  01:33:41
nve10      7000     L2CP 10.2.2.20        6              7000       UP   N/A  01:33:41
```

VTEP 2

The following example shows the output for the **show nve peers** command on VTEP 2:

```
VTEP2# show nve peers
Interface  VNI      Type Peer-IP          RMAC/Num_RTs   eVNI     state flags UP time
nve10      5000     L3CP 10.1.1.10        a0f8.4910.bce2 5000       UP  A/M/4 01:33:55
nve10      5000     L3CP 10.1.1.10        a0f8.4910.bce2 5000       UP  A/-/6 01:14:23
nve10      6000     L2CP 10.1.1.10        7              6000       UP   N/A  01:33:55
nve10      7000     L2CP 10.1.1.10        6              7000       UP   N/A  01:33:55
```

show l2vpn evpn peers vxlan

VTEP 1

The following example shows the output for the **show l2vpn evpn peers vxlan** command on VTEP 1:

```
VTEP1# l2vpn evpn peers vxlan
Interface VNI      Peer-IP                  Num routes eVNI     UP time
--------- -------- ------------------------ ---------- -------- --------
nve10     6000     10.2.2.20                5          6000     01:34:50
nve10     7000     10.2.2.20                6          7000     01:34:50
```

VTEP 2

The following example shows the output for the **show l2vpn evpn peers vxlan** command on VTEP 2:

```
VTEP2# show l2vpn evpn peers vxlan
Interface VNI      Peer-IP                  Num routes eVNI     UP time
--------- -------- ------------------------ ---------- -------- --------
nve10     6000     10.1.1.10                7          6000     01:35:23
nve10     7000     10.1.1.10                6          7000     01:35:23
```

show l2vpn evpn evi evpn-instance detail

VTEP 1

The following example shows the output for the **show l2vpn evpn evi** *evpn-instance* **detail** command on VTEP 1:

```
VTEP1# show l2vpn evpn evi 1 detail
EVPN instance:      1 (VLAN Based)
  RD:               10.1.1.10:1 (auto)
  Import-RTs:       10:1
  Export-RTs:       10:1
  Per-EVI Label:    none
  State:            Established
  Replication Type: Static (global)
  Encapsulation:    vxlan
  IP Local Learn:   Enable (global)
  Vlan:             201
    Ethernet-Tag:   0
    State:          Established
    Core If:        Vlan200
    Access If:      Vlan201
    NVE If:         nve10
    RMAC:           a0f8.4910.bce2
    Core Vlan:      200
    L2 VNI:         6000
    L3 VNI:         5000
    VTEP IP:        10.1.1.10
    MCAST IP:       232.1.1.1
    VRF:            green
    IPv4 IRB:       Enabled
    IPv6 IRB:       Enabled
    Pseudoports:
      GigabitEthernet1/0/1 service instance 201
```

VTEP 2

The following example shows the output for the **show l2vpn evpn evi** *evpn-instance* **detail** command on VTEP 2:

```
VTEP2# show l2vpn evpn evi 1 detail
EVPN instance:      1 (VLAN Based)
  RD:               10.2.2.20:1 (auto)
  Import-RTs:       10:1
  Export-RTs:       10:1
  Per-EVI Label:    none
  State:            Established
  Replication Type: Static (global)
  Encapsulation:    vxlan
  IP Local Learn:   Enable (global)
  Vlan:             201
    Ethernet-Tag:   0
    State:          Established
    Core If:        Vlan200
    Access If:      Vlan201
    NVE If:         nve10
    RMAC:           380e.4d9b.6a4a
    Core Vlan:      200
    L2 VNI:         6000
    L3 VNI:         5000
    VTEP IP:        10.2.2.20
    MCAST IP:       232.1.1.1
    VRF:            green
    IPv4 IRB:       Enabled
    IPv6 IRB:       Enabled
    Pseudoports:
      GigabitEthernet1/0/1 service instance 201
```

show l2vpn evpn default-gateway

VTEP 1

The following example shows the output for the **show l2vpn evpn default-gateway** command on VTEP 1:

```
VTEP1# show l2vpn evpn default-gateway
Valid Default Gateway Address   EVI   VLAN  MAC Address    Source
----- ------------------------- ----- ----- -------------- --------------------
  Y   192.168.1.201             1     201   a0f8.4910.bccc Vl201
  Y   192.168.1.201             1     201   380e.4d9b.6a48 10.2.2.20
  Y   2001:DB8:201::201         1     201   a0f8.4910.bccc Vl201
  Y   2001:DB8:201::201         1     201   380e.4d9b.6a48 10.2.2.20
  Y   192.168.2.202             2     202   a0f8.4910.bcc2 Vl202
  Y   192.168.2.202             2     202   380e.4d9b.6a42 10.2.2.20
  Y   2001:DB8:202::202         2     202   a0f8.4910.bcc2 Vl202
  Y   2001:DB8:202::202         2     202   380e.4d9b.6a42 10.2.2.20
```

VTEP 2

The following example shows the output for the **show l2vpn evpn default-gateway** command on VTEP 2:

```
VTEP2# show l2vpn evpn default-gateway
Valid Default Gateway Address   EVI   VLAN  MAC Address    Source
----- ------------------------- ----- ----- -------------- --------------------
  Y   192.168.1.201             1     201   380e.4d9b.6a48 Vl201
  Y   192.168.1.201             1     201   a0f8.4910.bccc 10.1.1.10
  Y   2001:DB8:201::201         1     201   380e.4d9b.6a48 Vl201
  Y   2001:DB8:201::201         1     201   a0f8.4910.bccc 10.1.1.10
  Y   192.168.2.202             2     202   380e.4d9b.6a42 Vl202
  Y   192.168.2.202             2     202   a0f8.4910.bcc2 10.1.1.10
  Y   2001:DB8:202::202         2     202   380e.4d9b.6a42 Vl202
  Y   2001:DB8:202::202         2     202   a0f8.4910.bcc2 10.1.1.10
```

show bgp l2vpn evpn all

VTEP 1

The following example shows the output for the **show bgp l2vpn evpn all** command on VTEP 1:

```
VTEP1# show bgp l2vpn evpn all
BGP table version is 705, local router ID is 10.1.1.10
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path, L long-lived-stale,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 10.1.1.10:1
 *>i  [2][10.1.1.10:1][0][48][0018736C56C3][0][*]/20
                      10.2.2.20                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][0018736C56C3][32][192.168.1.89]/24
                      10.2.2.20                0    100      0 ?
 *>   [2][10.1.1.10:1][0][48][0059DC50AE01][0][*]/20
                      ::                                 32768 ?
 *>   [2][10.1.1.10:1][0][48][0059DC50AE4C][0][*]/20
                      ::                                 32768 ?
 *>   [2][10.1.1.10:1][0][48][0059DC50AE4C][32][192.168.1.81]/24
                      ::                                 32768 ?
 *>   [2][10.1.1.10:1][0][48][0059DC50AE4C][128][2001:DB8:201::81]/36
                      ::                                 32768 ?
 *>   [2][10.1.1.10:1][0][48][0059DC50AE4C][128][FE80::259:DCFF:FE50:AE4C]/36
                      ::                                 32768 ?
 *>i  [2][10.1.1.10:1][0][48][380E4D9B6A48][32][192.168.1.201]/24
                      10.2.2.20                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][380E4D9B6A48][128][2001:DB8:201::201]/36
                      10.2.2.20                0    100      0 ?
 *>   [2][10.1.1.10:1][0][48][A0F84910BCCC][32][192.168.1.201]/24
                      ::                                 32768 ?
 *>   [2][10.1.1.10:1][0][48][A0F84910BCCC][128][2001:DB8:201::201]/36
                      ::                                 32768 ?
Route Distinguisher: 10.1.1.10:2
 *>i  [2][10.1.1.10:2][0][48][0018736C5681][0][*]/20
                      10.2.2.20                0    100      0 ?
 *>i  [2][10.1.1.10:2][0][48][0018736C56C2][0][*]/20
                      10.2.2.20                0    100      0 ?
 *>i  [2][10.1.1.10:2][0][48][0018736C56C2][32][192.168.2.89]/24
                      10.2.2.20                0    100      0 ?
 *>   [2][10.1.1.10:2][0][48][0059DC50AE01][0][*]/20
                      ::                                 32768 ?
 *>   [2][10.1.1.10:2][0][48][0059DC50AE42][0][*]/20
```

```
                        ::                                       32768 ?
 *>   [2][10.1.1.10:2][0][48][0059DC50AE42][32][192.168.2.81]/24
                        ::                                       32768 ?
 *>i  [2][10.1.1.10:2][0][48][380E4D9B6A42][32][192.168.2.202]/24
                        10.2.2.20                0    100      0 ?
 *>i  [2][10.1.1.10:2][0][48][380E4D9B6A42][128][2001:DB8:202::202]/36
                        10.2.2.20                0    100      0 ?
 *>   [2][10.1.1.10:2][0][48][A0F84910BCC2][32][192.168.2.202]/24
                        ::                                       32768 ?
 *>   [2][10.1.1.10:2][0][48][A0F84910BCC2][128][2001:DB8:202::202]/36
                        ::                                       32768 ?
Route Distinguisher: 10.2.2.20:1
 *>i  [2][10.2.2.20:1][0][48][0018736C56C3][0][*]/20
                        10.2.2.20                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][0018736C56C3][32][192.168.1.89]/24
                        10.2.2.20                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][380E4D9B6A48][32][192.168.1.201]/24
                        10.2.2.20                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][380E4D9B6A48][128][2001:DB8:201::201]/36
                        10.2.2.20                0    100      0 ?
Route Distinguisher: 10.2.2.20:2
 *>i  [2][10.2.2.20:2][0][48][0018736C5681][0][*]/20
                        10.2.2.20                0    100      0 ?
 *>i  [2][10.2.2.20:2][0][48][0018736C56C2][0][*]/20
                        10.2.2.20                0    100      0 ?
 *>i  [2][10.2.2.20:2][0][48][0018736C56C2][32][192.168.2.89]/24
                        10.2.2.20                0    100      0 ?
 *>i  [2][10.2.2.20:2][0][48][380E4D9B6A42][32][192.168.2.202]/24
                        10.2.2.20                0    100      0 ?
 *>i  [2][10.2.2.20:2][0][48][380E4D9B6A42][128][2001:DB8:202::202]/36
                        10.2.2.20                0    100      0 ?
Route Distinguisher: 103:2 (default for vrf green)
 *>   [5][103:2][0][24][10.1.13.0]/17
                        0.0.0.0                  0         32768 ?
 *>   [5][103:2][0][24][192.168.1.0]/17
                        0.0.0.0                  0         32768 ?
 *>   [5][103:2][0][24][192.168.2.0]/17
                        0.0.0.0                  0         32768 ?
 *>   [5][103:2][0][64][2001:DB8:201::]/29
                        ::                       0         32768 ?
 *>   [5][103:2][0][64][2001:DB8:202::]/29
                        ::                       0         32768 ?
Route Distinguisher: 104:2
 *>i  [5][104:2][0][24][10.1.14.0]/17
                        10.2.2.20                0    100      0 ?
 *>i  [5][104:2][0][24][192.168.1.0]/17
                        10.2.2.20                0    100      0 ?
 *>i  [5][104:2][0][24][192.168.2.0]/17
                        10.2.2.20                0    100      0 ?
 *>i  [5][104:2][0][64][2001:DB8:201::]/29
                        10.2.2.20                0    100      0 ?
 *>i  [5][104:2][0][64][2001:DB8:202::]/29
                        10.2.2.20                0    100      0 ?
```

VTEP 2

The following example shows the output for the **show bgp l2vpn evpn all** command on VTEP 2:

```
VTEP2# show bgp l2vpn evpn all
BGP table version is 584, local router ID is 10.2.2.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
```

```
              x best-external, a additional-path, c RIB-compressed,
              t secondary path, L long-lived-stale,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 10.1.1.10:1
 *>i  [2][10.1.1.10:1][0][48][0059DC50AE01][0][*]/20
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][0059DC50AE4C][0][*]/20
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][0059DC50AE4C][32][192.168.1.81]/24
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][0059DC50AE4C][128][2001:DB8:201::81]/36
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][0059DC50AE4C][128][FE80::259:DCFF:FE50:AE4C]/36
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][A0F84910BCCC][32][192.168.1.201]/24
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:1][0][48][A0F84910BCCC][128][2001:DB8:201::201]/36
                      10.1.1.10                0    100      0 ?
Route Distinguisher: 10.1.1.10:2
 *>i  [2][10.1.1.10:2][0][48][0059DC50AE01][0][*]/20
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:2][0][48][0059DC50AE42][0][*]/20
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:2][0][48][0059DC50AE42][32][192.168.2.81]/24
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:2][0][48][A0F84910BCC2][32][192.168.2.202]/24
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.1.1.10:2][0][48][A0F84910BCC2][128][2001:DB8:202::202]/36
                      10.1.1.10                0    100      0 ?
Route Distinguisher: 10.2.2.20:1
 *>   [2][10.2.2.20:1][0][48][0018736C56C3][0][*]/20
                      ::                                    32768 ?
 *>   [2][10.2.2.20:1][0][48][0018736C56C3][32][192.168.1.89]/24
                      ::                                    32768 ?
 *>i  [2][10.2.2.20:1][0][48][0059DC50AE01][0][*]/20
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][0059DC50AE4C][0][*]/20
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][0059DC50AE4C][32][192.168.1.81]/24
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][0059DC50AE4C][128][2001:DB8:201::81]/36
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][0059DC50AE4C][128][FE80::259:DCFF:FE50:AE4C]/36
                      10.1.1.10                0    100      0 ?
 *>   [2][10.2.2.20:1][0][48][380E4D9B6A48][32][192.168.1.201]/24
                      ::                                    32768 ?
 *>   [2][10.2.2.20:1][0][48][380E4D9B6A48][128][2001:DB8:201::201]/36
                      ::                                    32768 ?
 *>i  [2][10.2.2.20:1][0][48][A0F84910BCCC][32][192.168.1.201]/24
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.2.2.20:1][0][48][A0F84910BCCC][128][2001:DB8:201::201]/36
                      10.1.1.10                0    100      0 ?
Route Distinguisher: 10.2.2.20:2
 *>   [2][10.2.2.20:2][0][48][0018736C5681][0][*]/20
                      ::                                    32768 ?
 *>   [2][10.2.2.20:2][0][48][0018736C56C2][0][*]/20
                      ::                                    32768 ?
 *>   [2][10.2.2.20:2][0][48][0018736C56C2][32][192.168.2.89]/24
                      ::                                    32768 ?
 *>i  [2][10.2.2.20:2][0][48][0059DC50AE01][0][*]/20
                      10.1.1.10                0    100      0 ?
```

```
 *>i  [2][10.2.2.20:2][0][48][0059DC50AE42][0][*]/20
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.2.2.20:2][0][48][0059DC50AE42][32][192.168.2.81]/24
                      10.1.1.10                0    100      0 ?
 *>   [2][10.2.2.20:2][0][48][380E4D9B6A42][32][192.168.2.202]/24
                      ::                                 32768 ?
 *>   [2][10.2.2.20:2][0][48][380E4D9B6A42][128][2001:DB8:202::202]/36
                      ::                                 32768 ?
 *>i  [2][10.2.2.20:2][0][48][A0F84910BCC2][32][192.168.2.202]/24
                      10.1.1.10                0    100      0 ?
 *>i  [2][10.2.2.20:2][0][48][A0F84910BCC2][128][2001:DB8:202::202]/36
                      10.1.1.10                0    100      0 ?
Route Distinguisher: 103:2
 *>i  [5][103:2][0][24][10.1.13.0]/17
                      10.1.1.10                0    100      0 ?
 *>i  [5][103:2][0][24][192.168.1.0]/17
                      10.1.1.10                0    100      0 ?
 *>i  [5][103:2][0][24][192.168.2.0]/17
                      10.1.1.10                0    100      0 ?
 *>i  [5][103:2][0][64][2001:DB8:201::]/29
                      10.1.1.10                0    100      0 ?
 *>i  [5][103:2][0][64][2001:DB8:202::]/29
                      10.1.1.10                0    100      0 ?
Route Distinguisher: 104:2 (default for vrf green)
 *>   [5][104:2][0][24][10.1.14.0]/17
                      0.0.0.0                  0         32768 ?
 *>   [5][104:2][0][24][192.168.1.0]/17
                      0.0.0.0                  0         32768 ?
 *>   [5][104:2][0][24][192.168.2.0]/17
                      0.0.0.0                  0         32768 ?
 *>   [5][104:2][0][64][2001:DB8:201::]/29
                      ::                       0         32768 ?
 *>   [5][104:2][0][64][2001:DB8:202::]/29
                      ::                       0         32768 ?
```

show ip route vrf green

VTEP 1

The following example shows the output for the **show ip route vrf** *vrf-name* command on VTEP 1:

```
VTEP1# show ip route vrf green
Routing Table: green
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C        10.1.13.0/24 is directly connected, Loopback13
L        10.1.13.13/32 is directly connected, Loopback13
B        10.1.14.0/24 [200/0] via 10.2.2.20, 01:30:02, Vlan200
      192.168.1.0/24 is variably subnetted, 3 subnets, 2 masks
```

```
C        192.168.1.0/24 is directly connected, Vlan201
B        192.168.1.89/32 [200/0] via 10.2.2.20, 00:04:05, Vlan200
L        192.168.1.201/32 is directly connected, Vlan201
      192.168.2.0/24 is variably subnetted, 3 subnets, 2 masks
C        192.168.2.0/24 is directly connected, Vlan202
B        192.168.2.89/32 [200/0] via 10.2.2.20, 00:04:10, Vlan200
L        192.168.2.202/32 is directly connected, Vlan202
```

VTEP 2

The following example shows the output for the **show ip route vrf** *vrf-name* command on VTEP 2:

```
VTEP2# show ip route vrf green
Routing Table: green
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
B        10.1.13.0/24 [200/0] via 10.1.1.10, 01:31:17, Vlan200
C        10.1.14.0/24 is directly connected, Loopback14
L        10.1.14.14/32 is directly connected, Loopback14
      192.168.1.0/24 is variably subnetted, 3 subnets, 2 masks
C        192.168.1.0/24 is directly connected, Vlan201
B        192.168.1.81/32 [200/0] via 10.1.1.10, 01:39:53, Vlan200
L        192.168.1.201/32 is directly connected, Vlan201
      192.168.2.0/24 is variably subnetted, 3 subnets, 2 masks
C        192.168.2.0/24 is directly connected, Vlan202
B        192.168.2.81/32 [200/0] via 10.1.1.10, 01:39:30, Vlan200
L        192.168.2.202/32 is directly connected, Vlan202
```

show platform software fed switch active matm mactable vlan

VTEP 1

The following examples show the output for the **show platform software fed switch active matm mactable vlan** *vlan-id* command on VTEP 1:

Note The MAC address of the peer's core SVI interface must be present in the core VLAN.

```
VTEP1# show platform software fed switch active matm macTable vlan 200
VLAN   MAC                   Type  Seq#   EC_Bi  Flags  machandle          siHandle
      riHandle            diHandle              *a_time  *e_time  ports
------------------------------------------------------------------------------------------
200    a0f8.4910.bce2      0x8002      0  19880     64  0x7f5d8503fd48      0x7f5d852b6d28
```

```
      0x0                 0x5234                      0        0  Vlan200

200    380e.4d9b.6a4a   0x1000001      0      0     64  0x7f5d855bfaa8      0x7f5d852aca68
      0x7f5d851c7078      0x0                        0        0  RLOC 10.2.2.20 adj_id 126


Total Mac number of addresses:: 2

VTEP1# show platform software fed switch active matm macTable vlan 201
VLAN   MAC                   Type  Seq#   EC_Bi  Flags  machandle           siHandle
      riHandle            diHandle              *a_time  *e_time  ports
-----------------------------------------------------------------------------------------

201    00aa.00bb.00cc      0x8002      0  42949     64  0x7f5d85007b88      0x7f5d852b6d28
      0x0                 0x0                         0        0  Vlan201

201    0059.dc50.ae01         0x1      9      0      0  0x7f5d852abaf8      0x7f5d85035248
      0x0                 0x7f5d8517eae8            300        9  GigabitEthernet1/0/1

201    a0f8.4910.bccc      0x8002      0  19880     64  0x7f5d852ad618      0x7f5d852b6d28
      0x0                 0x5234                      0        9  Vlan201

201    0059.dc50.ae4c         0x1     16      0      0  0x7f5d855b3ff8      0x7f5d855a2858
      0x0                 0x7f5d8517eae8            300       95  GigabitEthernet1/0/1

201    380e.4d9b.6a48      0x8002      0      0     64  0x7f5d84fbf948      0x7f5d852b6d28
      0x0                 0x5234                      0       95  Vlan201

201    0018.736c.56c3   0x1000001      0      0     64  0x7f5d855c8268      0x7f5d852368b8
      0x7f5d855c6098      0x0                         0       95  RLOC 10.2.2.20 adj_id 36


Total Mac number of addresses:: 6

VTEP1# show platform software fed switch active matm macTable vlan 202
VLAN   MAC                   Type  Seq#   EC_Bi  Flags  machandle           siHandle
      riHandle            diHandle              *a_time  *e_time  ports
-----------------------------------------------------------------------------------------

202    a0f8.4910.bcc2      0x8002      0  19880     64  0x7f5d8503d288      0x7f5d852b6d28
      0x0                 0x0                         0        0  Vlan202

202    0059.dc50.ae01         0x1     10      0      0  0x7f5d852ac8b8      0x7f5d852ac668
      0x0                 0x7f5d8517eae8            300       15  GigabitEthernet1/0/1

202    0018.736c.5681   0x1000001      0      0     64  0x7f5d855ba7a8      0x7f5d855b0c58
      0x7f5d8518dea8      0x0                        0       15  RLOC 10.2.2.20 adj_id 125

202    0059.dc50.ae42         0x1     17      0      0  0x7f5d8518e848      0x7f5d855a5258
      0x0                 0x7f5d8517eae8            300      225  GigabitEthernet1/0/1

202    380e.4d9b.6a42      0x8002      0      0     64  0x7f5d855a59a8      0x7f5d852b6d28
      0x0                 0x5234                      0      225  Vlan202

202    0018.736c.56c2   0x1000001      0      0     64  0x7f5d8523d2b8      0x7f5d855b0c58
      0x7f5d8518dea8      0x0                        0      225  RLOC 10.2.2.20 adj_id 125


Total Mac number of addresses:: 6
```

VTEP 2

The following examples show the output for the **show platform software fed switch active matm mactable vlan** *vlan-id* command on VTEP 2:

Note: The MAC address of the peer's core SVI interface must be present in the core VLAN.

```
VTEP2# show platform software fed switch active matm macTable vlan 200
VLAN   MAC                  Type  Seq#   EC_Bi  Flags  machandle           siHandle
      riHandle            diHandle              *a_time  *e_time  ports
------------------------------------------------------------------------------------------
200    380e.4d9b.6a4a      0x8002      0    128     64  0x7fa88557f3a8      0x7fa885574e38
      0x0                 0x5174                     0        0  Vlan200

200    a0f8.4910.bce2   0x1000001      0      0     64  0x7fa8859a3d38      0x7fa885947ba8
      0x7fa88598bfb8      0x0                        0        0  RLOC 10.1.1.10 adj_id 155

Total Mac number of addresses:: 2

VTEP2# show platform software fed switch active matm macTable vlan 201
VLAN   MAC                  Type  Seq#   EC_Bi  Flags  machandle           siHandle
      riHandle            diHandle              *a_time  *e_time  ports
------------------------------------------------------------------------------------------
201    380e.4d9b.6a48      0x8002      0  42949     64  0x7fa885970018      0x7fa885574e38
      0x0                 0x5174                     0        0  Vlan201

201    0059.dc50.ae01   0x1000001      0      0     64  0x7fa8849e1be8      0x7fa88598da48
      0x7fa88598e1f8      0x0                        0        0  RLOC 10.1.1.10 adj_id 153

201    0059.dc50.ae4c   0x1000001      0      0     64  0x7fa885993e68      0x7fa88598da48
      0x7fa88598e1f8      0x0                        0        0  RLOC 10.1.1.10 adj_id 153

201    a0f8.4910.bccc      0x8002      0      0     64  0x7fa8859acc48      0x7fa885574e38
      0x0                 0x5174                     0        0  Vlan201

201    0018.736c.56c3         0x1     68      0      0  0x7fa8859d3908      0x7fa88599e108
      0x0                 0x7fa884f079d8           300      247  GigabitEthernet1/0/1

Total Mac number of addresses:: 5

VTEP2# show platform software fed switch active matm macTable vlan 202
VLAN   MAC                  Type  Seq#   EC_Bi  Flags  machandle           siHandle
      riHandle            diHandle              *a_time  *e_time  ports
------------------------------------------------------------------------------------------
202    380e.4d9b.6a42      0x8002      0  19018     64  0x7fa885994cd8      0x7fa885574e38
      0x0                 0x5174                     0        0  Vlan202

202    0018.736c.5681         0x1      9      0      0  0x7fa88599c4e8      0x7fa88599c218
      0x0                 0x7fa884f079d8           300        7  GigabitEthernet1/0/1

202    0059.dc50.ae01   0x1000001      0      0     64  0x7fa8859a3098      0x7fa8859a2dc8
      0x7fa88599ee48      0x0                        0        7  RLOC 10.1.1.10 adj_id 154
```

```
202    0059.dc50.ae42   0x1000001      0      0      64  0x7fa8849e6b78      0x7fa8859a2dc8
      0x7fa88599ee48      0x0                          0        7  RLOC 10.1.1.10 adj_id 154

202    a0f8.4910.bcc2      0x8002      0      0      64  0x7fa88594ddb8      0x7fa885574e38
      0x0                 0x5174                        0        7  Vlan202

202    0018.736c.56c2         0x1     67      0       0  0x7fa8859d3488      0x7fa8859834f8
      0x0                 0x7fa884f079d8              300      267  GigabitEthernet1/0/1


Total Mac number of addresses:: 6
```

Verifying EVPN VXLAN Anycast Gateway

The following table lists the **show** commands that are used to verify EVPN VXLAN distributed anycast gateway:

Table 6: Commands to Verify EVPN VXLAN Distributed Anycast Gateway

Command	Purpose
show l2vpn evpn default-gateway	Displays the default gateway database.
show l2vpn l2route default-gateway	Displays the list of sent or received default gateway routes.
show mac address-table	Displays the list of MAC addresses received in default gateway routes that are installed as static MAC addresses for an SVI interface.

CHAPTER 5

Configuring DHCP Relay in BGP EVPN VXLAN Fabric

Restrictions for DHCP Relay in BGP EVPN VXLAN Fabric

- DHCP relay in a BGP EVPN VXLAN fabric is supported in the following scenarios only when VRF-Lite is configured on the border VTEP and the border VTEP is connected to the DHCP server through an external router.
  - DHCP client in the tenant VRF and DHCP server in the Layer 3 default VRF
  - DHCP client in the tenant VRF and DHCP server in a different tenant VRF
  - DHCP client in the tenant VRF and DHCP server in a non-default non-VXLAN VRF
- DHCPv6 relay is not supported.

Information About DHCP Relay in BGP EVPN VXLAN Fabric

Networks use DHCP relay to forward DHCP packets between host devices and a DHCP server. In a BGP EVPN VXLAN fabric, you can configure a VTEP as a relay agent to provide DCHP relay services in a multi-tenant VXLAN environment.

When a network uses DHCP relay, DHCP messages move through the same switch in both directions. DHCP relay generally uses the gateway IP address (GiAddr) for scope selection and DHCP response messages. In a BGP EVPN VXLAN fabric that has distributed IP anycast gateway enabled, DHCP messages can return to any switch that hosts the respective GiAddr.

Deploying DHCP relay in an EVPN VXLAN network requires a different method for scope selection and a unique IP address for each switch in the network. The unique Loopback interface for a switch becomes the GiAddr that a switch uses to respond to the correct switch. DHCP option 82, also referred to as DHCP option VPN, is used for scope selection based on the Layer 2 VNI.

In a multi-tenant EVPN environment, DHCP relay uses the following sub-options of option 82:

- **Sub-Option 151(0x97)—Virtual Subnet Selection:**

  The virtual subnet selection sub-option is used to convey VRF-related information to the DHCP server in an MPLS VPN and a VXLAN EVPN multi-tenant environment.

  RFC 6607 provides the definition for this sub-option.

- **Sub-Option 11(0xb)—Server ID Override**

  The server identifier or server ID override sub-option allows the DHCP relay agent to specify a new value for the server ID option. The DHCP server inserts this new value in the reply packet. This sub-option allows the DHCP relay agent to act as the actual DHCP server. The DHCP relay agent begins to receive all the renew requests instead of the DHCP server. The server ID override sub-option contains the incoming interface IP address. The DHCP client accesses the DHCP relay agent using the incoming interface IP address. The DHCP client uses this information to send all the renew and release request packets to the DHCP relay agent. The DHCP relay agent adds all the appropriate sub-options and then forwards the renew and release request packets to the original DHCP server.

  For this function, Cisco's proprietary implementation is sub-option 152(0x98). To implement the suboption and manage the function, run the **ip dhcp relay sub-option type cisco** command in global configuration mode on the VTEP that acts as the DHCP relay agent.

  RFC 5107 provides the definition for this sub-option.

- **Sub-Option 5(0x5)—Link Selection:**

  The link selection sub-option provides a mechanism to separate the subnet or link, on which the DHCP client resides, from the GiAddr. The DHCP server uses this mechanism to communicate with the DHCP relay agent. The DHCP relay agent sets the sub-option to the correct subscriber subnet. The DHCP server then uses this value to assign an IP address different from the GiAddr. The DHCP relay agent sets the GiAddr to its own IP address to ensure that it is possible to forward the DHCP messages over the network.

  For this function, Cisco's proprietary implementation is sub-option 150(0x96). To manage the function, run the **ip dhcp relay sub-option type cisco** command in global configuration mode on the VTEP that acts as the DHCP relay agent.

  RFC 3527 provides the definition for this sub-option.

DHCP Relay on VTEPs

DHCP relay is generally configured on the default gateway that faces the DHCP client. You can configure a VTEP as a DHCP relay agent in different ways to automate IP addressing. The configuration depends on whether the DHCP server is present in the same network, the same VRF, or a different VRF compared to the DHCP client. When the DHCP server and DHCP client are in different VRFs, traffic is forwarded across the tenant or VRF boundaries.

The following are the common DHCP relay deployment scenarios for a BGP EVPN VXLAN fabric:

1. DHCP client in the tenant VRF and DHCP server in the Layer 3 default VRF
2. DHCP client in the tenant VRF and DHCP server in the same tenant VRF
3. DHCP client in the tenant VRF and DHCP server in a different tenant VRF
4. DHCP client in the tenant VRF and DHCP server in a non-default non-VXLAN VRF

Note The deployment scenarios 1, 3, and 4 are supported only when VRF-Lite is configured on the border VTEP and the border VTEP is connected to the DHCP server through an external router.

How to Configure DHCP Relay in BGP EVPN VXLAN Fabric

You must configure EVPN VXLAN Layer 2 and Layer 3 overlay networks before configuring BGP EVPN VXLAN interworking with DHCP relay. See How to Configure EVPN VXLAN Integrated Routing and Bridging, on page 54 for detailed steps.

Perform the following set of procedures to configure BGP EVPN VLAN interworking with DHCP relay:

Configuring DHCP Relay on a VTEP

To configure DHCP relay on a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip dhcp relay information option vpn** **Example:** `Device(config)# ip dhcp relay information option vpn`	Enables the device to insert VPN suboptions into the DHCP relay agent information option in the messages forwarded to the DHCP server and sets the GiAddr on the outgoing interface towards the DHCP server.
Step 4	**ip dhcp relay information option** **Example:** `Device(config)# ip dhcp relay information option`	Enables the system to insert a DHCP relay agent information option in the messages forwarded to the DHCP server.
Step 5	**ip dhcp relay override** *gateway-ip-address* **link-selection** **Example:** `Device(config)# ip dhcp relay override giaddr link-selection`	Sets the gateway IP address as the IP address of the DHCP relay agent and configures the server to assign an IP address that is different from the GiAddr to the DHCP clients.

	Command or Action	Purpose
Step 6	**ip dhcp compatibility suboption server-override standard** **Example:** `Device(config)# ip dhcp compatibility suboption server-override standard`	Configures the DHCP client to use the Internet Assigned Numbers Authority (IANA) standard relay agent server ID override suboption.
Step 7	**ip dhcp snooping vlan** *vlan-id-list* **Example:** `Device(config)# ip dhcp snooping vlan 201-202`	Enables DHCP snooping on the specified list of VLANs.
Step 8	**ip dhcp snooping** **Example:** `Device(config)# ip dhcp snooping`	Enables DHCP snooping on the VTEP.
Step 9	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Configuring DHCP Relay on the Access SVI of a VTEP

Perform this procedure on all the VTEPs for each VLAN that is associated with the Layer 2 VNI configured in the EVPN VXLAN network.

To configure DHCP relay on the access SVI of a VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface vlan** *vlan-id* **Example:** `Device(config)# interface Vlan 201`	Enters interface configuration mode for the specified VLAN interface. This VLAN interface acts as the GiAddr.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding green`	Associates the VRF with the interface. The interface must be associated with the same VRF for which the Layer 3 VNI has been configured for the EVPN VXLAN network.

	Command or Action	Purpose
Step 5	**ip dhcp relay information option vpn-id** **Example:** `Device(config-if)# ip dhcp relay information option vpn-id`	Enables the device to insert VPN suboptions into the DHCP relay agent information option in the messages forwarded to the DHCP server and sets the GiAddr on the outgoing interface towards the DHCP server.
Step 6	**ip dhcp relay source-interface Loopback** *loopback-interface-id* **Example:** `Device(config-if)# ip dhcp relay source-interface Loopback13`	Configures the specified Loopback interface as the source interface for DHCP relay messages. The DHCP relay agent uses the IP address of the source interface as the source IP address to relay messages. **Note** The IP address configured on the Loopback interface must be unique per VTEP per VRF.
Step 7	**ip address** *ip-address* **Example:** `Device(config-if)# ip address 192.168.1.201 255.255.255.0`	Sets the IP address for the VLAN interface.
Step 8	**ip helper-address** *ip-address* **Example:** `Device(config-if)# ip helper-address 192.168.3.100`	Sets the DHCP IP helper address for the VLAN interface.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 10	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Configuring the Router Interface on the Border VTEP for DHCP Server Reachability

DHCP server reachability can be achieved through a physical Layer 3 interface or subinterface, or a Layer3 Portchannel interface.

To configure the router interface on the border VTEP for DHCP server rechability, perform the following steps:

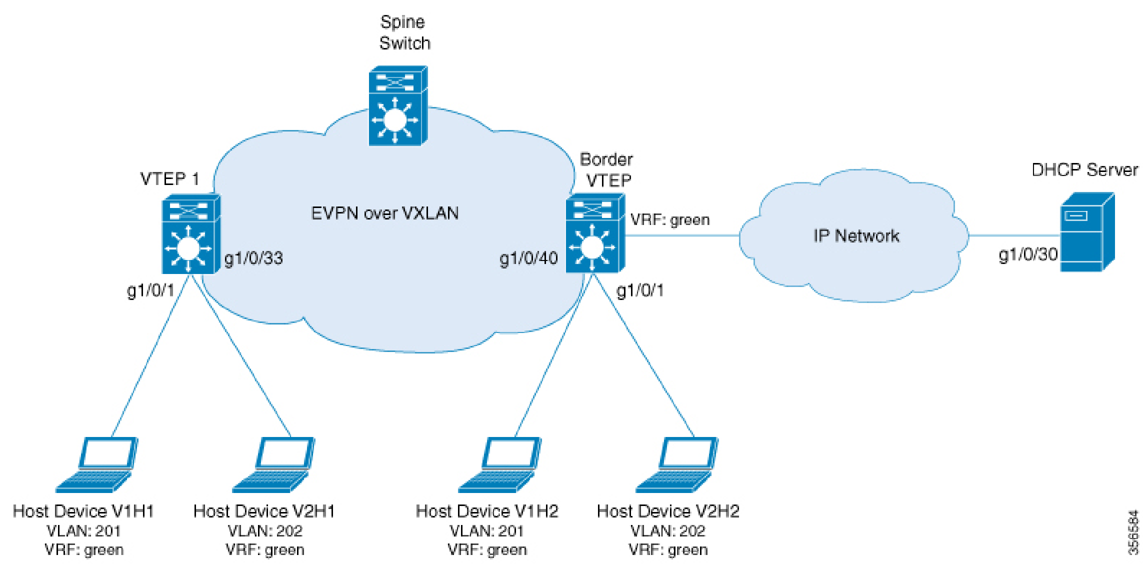
Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface vlan***vlan-id* **Example:** `Device(config)# interface vlan 203`	Enters interface configuration mode for the specified VLAN interface.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding green`	Configures the SVI for the VLAN and associates the specified VRF with the interface.
Step 5	**ip address** *ip-address* **Example:** `Device(config-if)# ip address 192.168.3.203 255.255.255.0`	Configures the IP address for the VLAN.
Step 6	**ipv6 address** *ipv6-address* **Example:** `Device(config-if)# ipv6 address 2001:203::203/64`	Configures the IPv6 address for the VLAN.
Step 7	**ipv6 enable** **Example:** `Device(config-if)# ipv6 enable`	Enables IPv6 processing on the VLAN interface.
Step 8	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 9	**interface** *interface-id* **Example:** `Device(config)# interface GigabitEthernet1/0/30`	Enters interface configuration mode for the specified interface.
Step 10	**switchport access vlan** *vlan-id* **Example:** `Device(config-if)# switchport access vlan 203`	Specifies the VLAN to be used as access VLAN when the interface is in access mode.

	Command or Action	Purpose
Step 11	**switchport mode access** **Example:** `Device(config-if)# switchport mode access`	Configures the interface as an access interface.
Step 12	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 13	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Configuration Examples for DHCP Relay in BGP EVPN VXLAN Fabric

This section provides an example to show the configuration and verification of DHCP relay deployment in an EVPN VXLAN network. The example uses the following topology where the DHCP client and the DHCP server are in the same tenant VRF:



The illustration shows an EVPN VXLAN network with two VTEPs, VTEP 1 and Border VTEP. Border VTEP is connected to the DHCP server.

DHCP server reachability can be achieved through a physical Layer 3 interface or subinterface, or a Layer3 Portchannel interface. The example shown here deploys DHCP relay using an SVI interface and a switchport.

Table 7: Configuration Example for Deploying DHCP Relay in a BGP EVPN VXLAN Fabric when the DHCP Client and the DHCP Server are in the Same Tenant VRF

VTEP 1	Border VTEP

`VTEP1#` **`show running-config`**
 	
`<snip: only dhcp relevant config is shown>`
`ip dhcp relay information option vpn`
`ip dhcp relay information option`
`ip dhcp compatibility suboption link-selection standard`
`ip dhcp compatibility suboption server-override standard`
`ip dhcp snooping vlan 201-202`
`ip dhcp snooping`
`!`
`vlan configuration 200`
` member vni 5000`
`vlan configuration 201`
` member evpn-instance 1 vni 6000`
`vlan configuration 202`
` member evpn-instance 2 vni 7000`
`!`
`interface Loopback13`
` vrf forwarding green`
` ip address 10.1.13.13 255.255.255.0`
 	
`interface Vlan200`
` description core svi for l3vni`
` vrf forwarding green`
` ip unnumbered Loopback0`
` ip pim sparse-mode`
` ipv6 enable`
` no autostate`
 	
`interface Vlan201`
` vrf forwarding green`
` ip dhcp relay information option vpn-id`
` ip dhcp relay source-interface Loopback13`
` ip address 192.168.1.201 255.255.255.0`
` ip helper-address 192.168.3.100`
 	
`interface Vlan202`
` vrf forwarding green`
` ip dhcp relay information option vpn-id`
` ip dhcp relay source-interface Loopback13`
` ip address 192.168.2.201 255.255.255.0`
` ip helper-address 192.168.3.100`
 	
`interface nve10`
` no ip address`
` source-interface Loopback0`
` host-reachability protocol bgp`
` member vni 7000 mcast-group 231.1.1.1`
` member vni 6000 mcast-group 231.1.1.1`
` member vni 5000 vrf green`
	`Border_VTEP#` **`show running-config`**
	
	`<snip: only dhcp relevant config is shown>`
	`ip dhcp relay information option vpn`
	`ip dhcp relay information option`
	`ip dhcp relay override giaddr link-selection`
	`ip dhcp compatibility suboption server-override standard`
	`ip dhcp snooping vlan 201-202`
	`ip dhcp snooping`
	`!`
	`vlan configuration 200`
	` member vni 5000`
	`vlan configuration 201`
	` member evpn-instance 1 vni 6000`
	`vlan configuration 202`
	` member evpn-instance 2 vni 7000`
	`!`
	`interface Loopback14`
	` vrf forwarding green`
	` ip address 10.1.14.14 255.255.255.0`
	
	`interface Vlan200`
	` description core svi for l3vni`
	` vrf forwarding green`
	` ip unnumbered Loopback0`
	` ip pim sparse-mode`
	` ipv6 enable`
	` no autostate`
	
	`interface Vlan201`
	` vrf forwarding green`
	` ip dhcp relay information option vpn-id`
	` ip dhcp relay source-interface Loopback14`
	` ip address 192.168.1.201 255.255.255.0`
	` ip helper-address 192.168.3.100`
	
	`interface Vlan202`
	` vrf forwarding green`
	` ip dhcp relay information option vpn-id`
	` ip dhcp relay source-interface Loopback14`
	` ip address 192.168.2.201 255.255.255.0`
	` ip helper-address 192.168.3.100`
	
	`interface nve10`
	` no ip address`
	` source-interface Loopback0`
	` host-reachability protocol bgp`
	` member vni 7000 mcast-group 231.1.1.1`
	` member vni 6000 mcast-group 231.1.1.1`
	` member vni 5000 vrf green`

<table>
<tr><th>VTEP 1</th><th>Border VTEP</th></tr>
<tr><td>As VTEP 1 is not a border VTEP, DHCP server reachability is not configured on VTEP 1.</td><td><pre>
interface Vlan203
 vrf forwarding green
 ip address 192.168.3.203 255.255.255.0
 ipv6 address 2001:203::203/64
 ipv6 enable
end

interface GigabitEthernet1/0/30
 description connected to DHCP server
 switchport access vlan 203
 switchport mode access
</pre></td></tr>
</table>

The following examples provide sample outputs for the **show ip route vrf** command on VTEP 1 and Border VTEP to verify the reachability of the DHCP server from both VTEPs:

VTEP 1

The following example shows the output for the **show ip route vrf** command on VTEP 1:

```
VTEP1# show ip route vrf green 192.168.3.100

Routing Table: green
Routing entry for 192.168.3.0/24
  Known via "bgp 10", distance 200, metric 0, type internal
  Last update from 10.2.2.20 on Vlan200, 18:28:43 ago
  Routing Descriptor Blocks:
  * 10.2.2.20 (default), from 10.5.5.50, 18:28:43 ago, via Vlan200
      opaque_ptr 0x7FEEA41D09C8
      Route metric is 0, traffic share count is 1
      AS Hops 0
      MPLS label: none
      MPLS Flags: NSF
```

Border VTEP

The following example shows the output for the **show ip route vrf** command on VTEP 2:

```
Border_VTEP# show ip route vrf green 192.168.3.100

Routing Table: green
Routing entry for 192.168.3.0/24
  Known via "connected", distance 0, metric 0 (connected, via interface)
  Redistributing via bgp 10
  Advertised by bgp 10
  Routing Descriptor Blocks:
  * directly connected, via Vlan203
      Route metric is 0, traffic share count is 1
```

Packet Capture for Spine Switch

The following example shows the packet capture details for the spine switch from the topology configured above:

```
    6  12.749326   10.1.13.13 b^F^R 192.168.3.100 DHCP 449 DHCP Discover - Transaction ID
0x228f
    7  12.750463 192.168.3.100 b^F^R 10.1.13.13   DHCP 447 DHCP Offer    - Transaction ID
0x228f
    8  12.755776   10.1.13.13 b^F^R 192.168.3.100 DHCP 467 DHCP Request  - Transaction ID
0x228f
    9  12.756701 192.168.3.100 b^F^R 10.1.13.13   DHCP 447 DHCP ACK      - Transaction ID
0x228f
   11  12.803031 00:59:dc:50:ae:42 b^F^R ff:ff:ff:ff:ff:ff ARP 110 Gratuitous ARP for
192.168.2.3 (Reply)
   14  15.760480 00:59:dc:50:ae:42 b^F^R ff:ff:ff:ff:ff:ff ARP 110 Who has 192.168.2.201?
Tell 192.168.2.3
   15  15.761058 38:0e:4d:9b:6a:42 b^F^R 00:59:dc:50:ae:42 ARP 110 192.168.2.201 is at
38:0e:4d:9b:6a:42
```

Discover Packet Details for VTEP 1

The following example shows the packet discovery details for VTEP 1 from the topology configured above:

```
Frame 6: 449 bytes on wire (3592 bits), 449 bytes captured (3592 bits) on interface 0
    Interface id: 0 (/tmp/epc_ws/wif_to_ts_pipe)
        Interface name: /tmp/epc_ws/wif_to_ts_pipe
    Encapsulation type: Ethernet (1)
    Arrival Time: Mar 28, 2020 09:03:26.742700000 UTC
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1585386206.742700000 seconds
    [Time delta from previous captured frame: 7.090744000 seconds]
    [Time delta from previous displayed frame: 7.090744000 seconds]
    [Time since reference or first frame: 12.749326000 seconds]
    Frame Number: 6
    Frame Length: 449 bytes (3592 bits)
    Capture Length: 449 bytes (3592 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:udp:vxlan:eth:ethertype:ip:udp:bootp]
Ethernet II, Src: 00:a3:d1:5a:03:61 (00:a3:d1:5a:03:61), Dst: 38:0e:4d:9b:6a:45
(38:0e:4d:9b:6a:45)
    Destination: 38:0e:4d:9b:6a:45 (38:0e:4d:9b:6a:45)
        Address: 38:0e:4d:9b:6a:45 (38:0e:4d:9b:6a:45)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: 00:a3:d1:5a:03:61 (00:a3:d1:5a:03:61)
        Address: 00:a3:d1:5a:03:61 (00:a3:d1:5a:03:61)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.1.1.10, Dst: 10.2.2.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        0000 00.. = Differentiated Services Codepoint: Default (0)
        .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 435
    Identification: 0xc29c (49820)
    Flags: 0x4000, Don't fragment
        0... .... .... .... = Reserved bit: Not set
        .1.. .... .... .... = Don't fragment: Set
        ..0. .... .... .... = More fragments: Not set
        ...0 0000 0000 0000 = Fragment offset: 0
```

```
        Time to live: 253
        Protocol: UDP (17)
        Header checksum: 0xa27c [validation disabled]
        [Header checksum status: Unverified]
        Source: 10.1.1.10
        Destination: 10.2.2.20
    User Datagram Protocol, Src Port: 65294, Dst Port: 4789
        Source Port: 65294
        Destination Port: 4789
        Length: 415
        [Checksum: [missing]]
        [Checksum Status: Not present]
        [Stream index: 0]
    Virtual eXtensible Local Area Network
        Flags: 0x0800, VXLAN Network ID (VNI)
            0... .... .... .... = GBP Extension: Not defined
            .... .... .0.. .... = Don't Learn: False
            .... 1... .... .... = VXLAN Network ID (VNI): True
            .... .... .... 0... = Policy Applied: False
            .000 .000 0.00 .000 = Reserved(R): 0x0000
        Group Policy ID: 0
        VXLAN Network Identifier (VNI): 5000
        Reserved: 0
    Ethernet II, Src: a0:f8:49:10:00:00 (a0:f8:49:10:00:00), Dst: 38:0e:4d:9b:6a:4a
    (38:0e:4d:9b:6a:4a)
        Destination: 38:0e:4d:9b:6a:4a (38:0e:4d:9b:6a:4a)
            Address: 38:0e:4d:9b:6a:4a (38:0e:4d:9b:6a:4a)
            .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
            .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        Source: a0:f8:49:10:00:00 (a0:f8:49:10:00:00)
            Address: a0:f8:49:10:00:00 (a0:f8:49:10:00:00)
            .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
            .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        Type: IPv4 (0x0800)
    Internet Protocol Version 4, Src: 10.1.13.13, Dst: 192.168.3.100
        0100 .... = Version: 4
        .... 0101 = Header Length: 20 bytes (5)
        Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
            0000 00.. = Differentiated Services Codepoint: Default (0)
            .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
        Total Length: 385
        Identification: 0x083f (2111)
        Flags: 0x0000
            0... .... .... .... = Reserved bit: Not set
            .0.. .... .... .... = Don't fragment: Not set
            ..0. .... .... .... = More fragments: Not set
            ...0 0000 0000 0000 = Fragment offset: 0
        Time to live: 254
        Protocol: UDP (17)
        Header checksum: 0xd812 [validation disabled]
        [Header checksum status: Unverified]
        Source: 10.1.13.13
        Destination: 192.168.3.100
    User Datagram Protocol, Src Port: 67, Dst Port: 67
        Source Port: 67
        Destination Port: 67
        Length: 365
        Checksum: 0x26ca [unverified]
        [Checksum Status: Unverified]
        [Stream index: 2]
    Bootstrap Protocol (Discover)
        Message type: Boot Request (1)
        Hardware type: Ethernet (0x01)
        Hardware address length: 6
```

```
Hops: 1
Transaction ID: 0x0000228f
Seconds elapsed: 0
Bootp flags: 0x8000, Broadcast flag (Broadcast)
    1... .... .... .... = Broadcast flag: Broadcast
    .000 0000 0000 0000 = Reserved flags: 0x0000
Client IP address: 0.0.0.0
Your (client) IP address: 0.0.0.0
Next server IP address: 0.0.0.0
Relay agent IP address: 10.1.13.13
Client MAC address: 00:59:dc:50:ae:42 (00:59:dc:50:ae:42)
Client hardware address padding: 00000000000000000000
Server host name not given
Boot file name not given
Magic cookie: DHCP
Option: (53) DHCP Message Type (Discover)
    Length: 1
    DHCP: Discover (1)
Option: (57) Maximum DHCP Message Size
    Length: 2
    Maximum DHCP Message Size: 1152
Option: (61) Client identifier
    Length: 27
    Type: 0
    Client Identifier: cisco-0059.dc50.ae42-Vl202
Option: (12) Host Name
    Length: 12
    Host Name: host-switch1
Option: (55) Parameter Request List
    Length: 8
    Parameter Request List Item: (1) Subnet Mask
    Parameter Request List Item: (6) Domain Name Server
    Parameter Request List Item: (15) Domain Name
    Parameter Request List Item: (44) NetBIOS over TCP/IP Name Server
    Parameter Request List Item: (3) Router
    Parameter Request List Item: (33) Static Route
    Parameter Request List Item: (150) TFTP Server Address
    Parameter Request List Item: (43) Vendor-Specific Information
Option: (60) Vendor class identifier
    Length: 8
    Vendor class identifier: ciscopnp
Option: (82) Agent Information Option
    Length: 44
    Option 82 Suboption: (1) Agent Circuit ID
        Length: 12
        Agent Circuit ID: 010a000800001b5801010000
    Option 82 Suboption: (2) Agent Remote ID
        Length: 8
        Agent Remote ID: 0006a0f84910bc80
    Option 82 Suboption: (151) VRF name/VPN ID
        Length: 6
        VRF name:
    Option 82 Suboption: (5) Link selection
        Length: 4
        Link selection: 192.168.2.0
    Option 82 Suboption: (11) Server ID Override
        Length: 4
        Server ID Override: 192.168.2.201
Option: (255) End
    Option End: 255
```

CHAPTER 6

Configuring EVPN VXLAN External Connectivity

Restrictions for EVPN VXLAN External Connectivity

- External connectivity through border spine switches is not supported.
- External connectivity with VPLS networks is supported only when bridging is the mode of interworking between the two domains. Integrated routing and bridging (IRB) is not supported between a BGP EVPN VXLAN fabric and a VPLS network.
- External Connectivity with Layer 3 networks is supported only for IPv4 and IPv6 unicast traffic.
- External connectivity with an MVPN network is not supported for multicast traffic.
- Import of EVPN IP routes, which includes both route type 5 and route type 2 host routes, to global routing table is not supported.

Information About EVPN VXLAN External Connectivity

External connectivity allows the movement of Layer 2 and Layer 3 traffic between an EVPN VXLAN network and an external network. It also enables the EVPN VXLAN network to exchange routes with the externally connected network. Routes within an EVPN VXLAN network are already shared between all the VTEPs or leaf switches. External connectivity uses the VTEPs on the periphery of the network to pass on these routes to an external Layer 2 or Layer 3 network. Similarly, the EVPN VXLAN network imports the reachability routes from the external network. External connectivity extends the Layer 2 or Layer 3 overlay network outside the VXLAN network. The process of extending a Layer 2 or Layer 3 network outside the EVPN VXLAN network is also known as handoff.

Implementation of Border Nodes for EVPN VXLAN External Connectivity

Border nodes or border VTEPs are the devices through which you establish a connection between an EVPN VXLAN network and an external network. The border nodes sit on the periphery of the EVPN VXLAN network and remain a part of the BGP EVPN VXLAN fabric. To enable external connectivity, you can implement the border nodes of an EVPN VXLAN network as either border leaf or border spine switches.

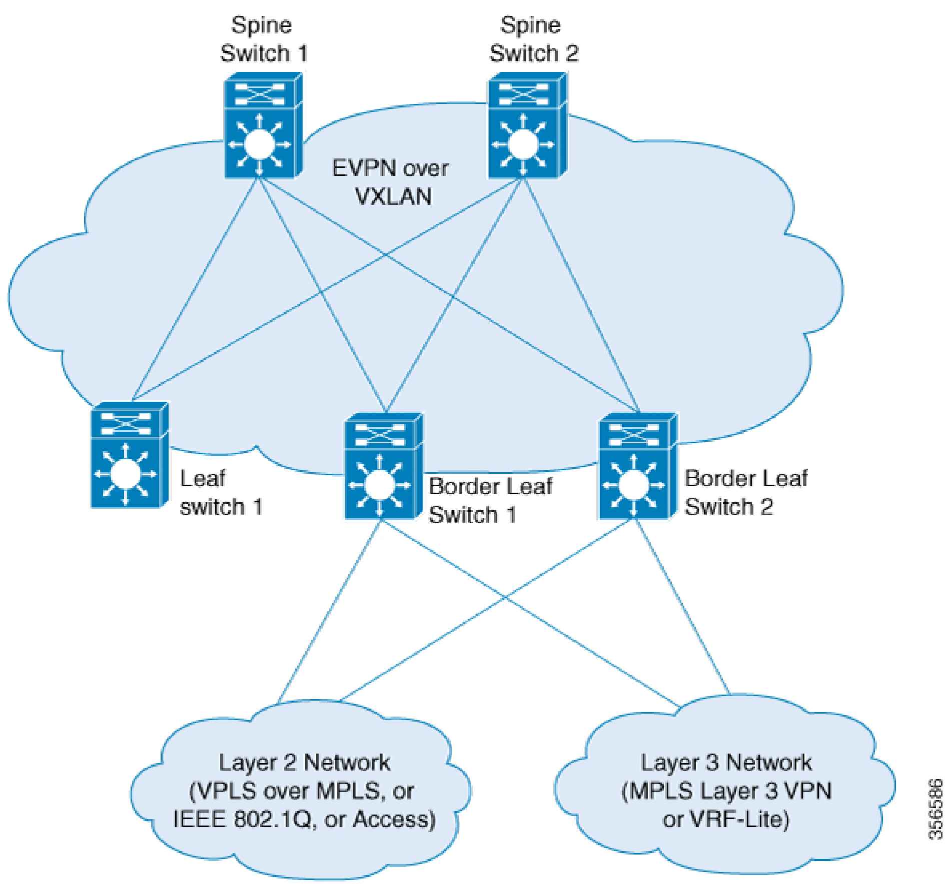
Connectivity Through a Border Leaf Switch

Leaf switches deployed as border nodes support the required control plane and data plane functionalities. Border leaf deployment ensures that the configuration on the spine switches is much simpler. Border leaf switches only allow communication between the external network and the VXLAN network, also known as north-south communication.

Note A border leaf switch can also be multiple switches functioning as a single logical system with Cisco StackWise Virtual configured.

The following figure shows border leaf external connectivity of an EVPN VXLAN network with external Layer 2 and Layer 3 networks.:

Figure 2: EVPN VXLAN External Connectivity Through a Border Leaf Switch

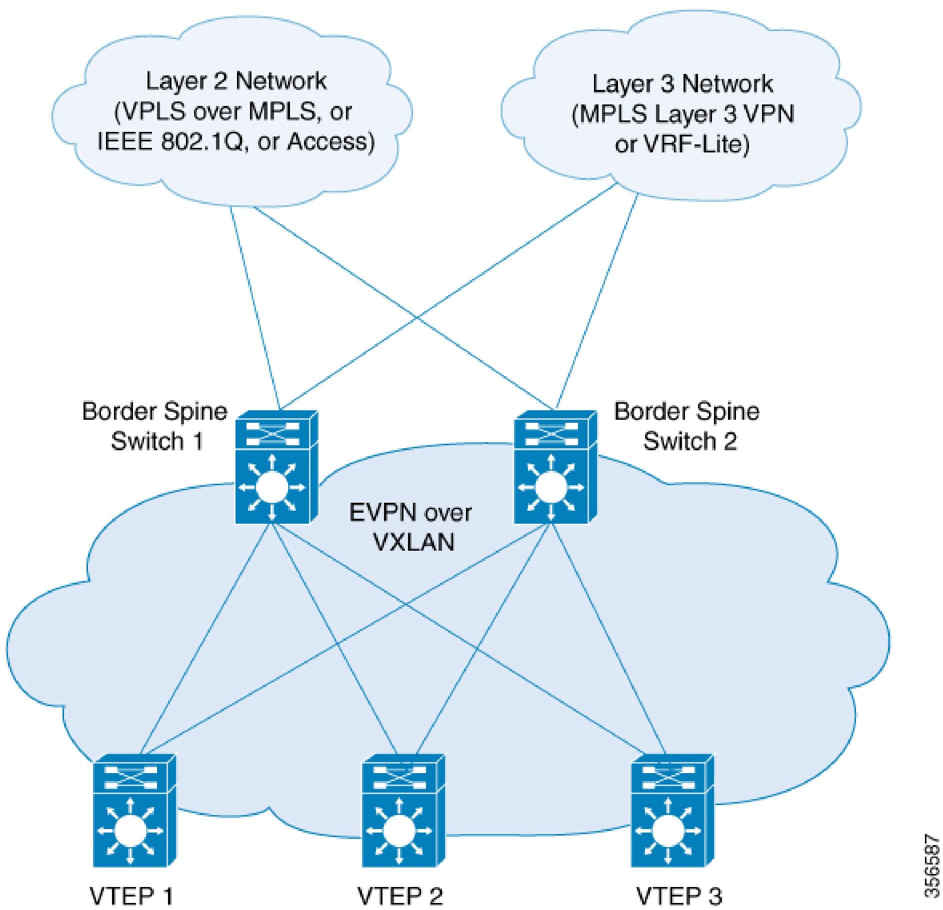


Connectivity Through a Border Spine Switch

Deploying spine switches as border nodes provides the advantage of optimizing the north-south communication with external resources. At the same time, border spine deployment allows the spine switches to support VXLAN control and data plane functionality. Border spine switches allow both north-south communication and east-west communication. East-west communication represents the communication within the nodes of the EVPN VXLAN network.

The following figure shows border spine external connectivity of an EVPN VXLAN network with external Layer 2 and Layer 3 networks.:

Figure 3: EVPN VXLAN External Connectivity Through a Border Spine Switch



External Connectivity with Layer 3 Networks

Layer 3 external connectivity or handoff is established by connecting the border nodes of a BGP EVPN VXLAN fabric with an edge router from the external Layer 3 network. The border node acts as a VTEP to perform VXLAN encapsulation and decapsulation, but it also routes the traffic towards the edge routing device. The VXLAN-facing interface on the external Layer 3 network can be a switch virtual interface (SVI), or a Layer 3 interface, or a Layer 3 subinterface.

You can use Layer 3 external connectivity to achieve any of the following:

- Extend the logical isolation between VRFs or VLANs within the EVPN VXLAN network into the externally routed network. The external routed network can be a traditional non-VXLAN campus network, a datacenter, or a WAN.
- Provide shared access within the EVPN VXLAN network to a common external service such as the internet.

BGP EVPN VXLAN fabric supports Layer 3 external connectivity with VRF-Lite and MPLS Layer 3 VPN networks.

Layer 3 External Connectivity with VRF-Lite

Using VRF allows for the use of multiple routing tables that are independent and isolated. VRF-Lite is a mechanism to extend the tenant Layer 3 VRF information beyond the BGP EVPN VXLAN Fabric. External connectivity with VRF-Lite or VRF handoff involves a two-box approach where the border node and the edge router are physically independent devices. With VRF-Lite handoff, the BGP EVPN VXLAN fabric extends the connectivity for different tenants externally on a hop-by-hop basis.

Once the border node learns external routes from the edge router, it advertises the prefixes inside the BGP EVPN VXLAN fabric as EVPN type 5 routes. This information is distributed to all the other VTEPs in the network. The border node also advertises EVPN routes to the external edge router. It sends the EVPN routes learned from the Layer 2 VPN EVPN address family to the IPv4 or IPv6 unicast address family.

Layer 3 Multicast External Connectivity with MPLS Layer 3 VPN

Layer 3 external connectivity with an MPLS Layer 3 VPN network or MPLS handoff uses a single-box approach. The single-box approach combines the functionalities of an EVPN VXLAN border node and an MPLS PE router into a single physical device. The device is also known as a border PE node. The border PE node reoriginates IP prefixes from the EVPN address family of the BGP EVPN VXLAN fabric to the VPNv4 address family of the MPLS network. Likewise, the border PE node performs the corresponding function in the reverse direction. eBGP peering is necessary between the border PE node and the MPLS PE devices to ensure the connectivity.

MPLS handoff allows scalability for EVPN VXLAN networks that have a large number of tenants or VRFs. Scalability is not possible with VRF-Lite handoff.

In every VRF on a border VTEP, there are two sets of manually configured import and export route targets. The first set of import and export route targets is associated with the BGP neighbor in the BGP EVPN VXLAN fabric. This BGP neighbor uses the EVPN address family to exchange Layer 3 information. The second set of import and export route targets is associated with the BGP neighbor in the Layer 3 VPN network. This BGP neighbor uses either VPNv4 or VPNv6 unicast address families to exchange Layer 3 information. The separation of route targets allows you to configure both sets of route targets independently. In this way, a border VTEP in an EVPN VXLAN network effectively stitches the two sets of route targets. The route targets associated with the BGP neighbor in the Layer 3 VPN network are known as normal route targets. The route targets associated with the BGP neighbor in the BGP EVPN VXLAN fabric are known as stitching route targets.

External Connectivity with Layer 2 Networks

Layer 2 external connectivity or handoff for an EVPN VXLAN network extends the Layer 2 domain outside of the network. BGP EVPN VXLAN fabric supports Layer 2 external connectivity with IEE 802.1Q, access, and VPLS over MPLS networks.

Layer 2 External connectivity with IEEE 802.1Q or Access Networks

Layer 2 handoff to IEEE 802.1Q networks is achieved through a regular IEEE 802.1Q Trunk port configuration on the Switchport interfaces on the border nodes. You can also connect EVPN VXLAN networks to external access networks.

The commonly deployed scenario has EVPN enabled at the distribution layer and has the access layer switches connected with IEEE 802.1Q Trunk encapsulation. The IEEE 802.1Q Layer 2 traffic that comes from the access layer switches is mapped to the corresponding VLAN. The border node then bridges the traffic towards the destination with VXLAN encapsulation. The inner packet does not carry the IEEE 802.1Q tag. Instead, the VXLAN network identifier (VNI), which is the Layer 2 VNI in the VXLAN header, represents the broadcast domain. Similarly, the border nodes decapsulate the traffic from the BGP EVPN VXLAN fabric and bridge it with the corresponding IEEE 802.1Q tag to the access switches. The interface on the border VTEP that faces the external interface can be either an access or a Trunk port. The external interface can belong to either a Layer 2 switch or a firewall.

Note If you connect the network to an external Layer 2 switch through two border VTEPs, it represents a dual connection. In such cases, STP does not propagate over the BGP EVPN VXLAN fabric by default.

Layer 2 External connectivity with VPLS over MPLS Network

External connectivity with VPLS networks or VPLS handoff is achieved when a border VTEP or multiple border VTEPs establish a connection with the VPLS network. The border nodes act as the provider edge (PE) devices in the VPLS network and as VTEPs in the EVPN VXLAN network.

BGP EVPN VXLAN supports VPLS handoff in the form of VPLS stitching through either an access VFI or an access pseudowire on the VLAN on the border VTEP.

The access pseudowires and the pseudowires in the access VFI function as the access ports in the EVPN VXLAN network. The BGP EVPN VXLAN fabric treats the MAC addresses learned on the pseudowires as locally learned MAC addresses. It advertises these MAC addresses within the fabric as EVPN type 2 routes. The pseudowires are in a different split horizon group compared to the EVPN VXLAN network. Therefore, BUM traffic floods between both the EVPN VXLAN and VPLS networks.

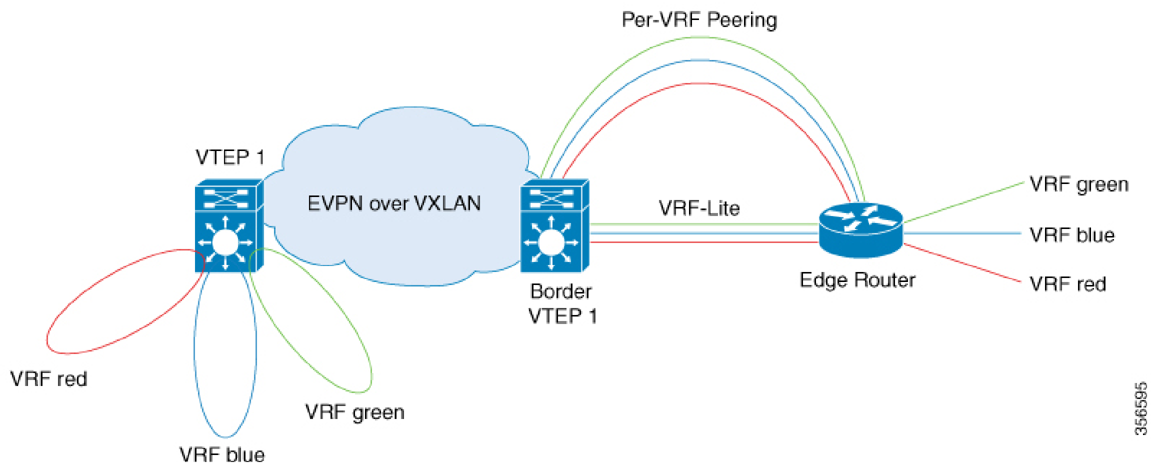
How to Configure EVPN VXLAN External Connectivity

This section provides information about how to configure external connectivity between an EVPN VXLAN network and an external Layer 2 or Layer 3 network.

Note You must configure EVPN VXLAN Layer 2 and Layer 3 overlay networks before you configure external connectivity. See How to Configure EVPN VXLAN Integrated Routing and Bridging, on page 54 for detailed steps.

Enabling Layer 3 External Connectivity with VRF-Lite

The following figure shows a sample topology that illustrates Layer 3 external connectivity with VRF-Lite:



To configure Layer 3 external connectivity with VRF-Lite, perform the following set of procedures:

- Configure the VRF on the border VTEP interface that faces the external router.
- Ensure that Layer 2 VPN EVPN is advertised as part of the BGP VRF configuration. See Configuring BGP with EVPN and VRF Address Families on a VTEP, on page 59 for detailed steps.

Note Redistribution of the respective interior gateway protocol (IGP) is required in the BGP VRF address family to distribute the external prefixes into the BGP EVPN VXLAN fabric.

For more information about VRF-Lite, see *Contents* → *IP Routing Configuration Guide* → *Configuring VRF-lite* in the software configuration guide for the applicable release.

Configuring the VRF on the Border VTEP Interface that Faces the External Router

To cnofigure the VRF on the border VTEP interface that faces the external router, perform these steps:

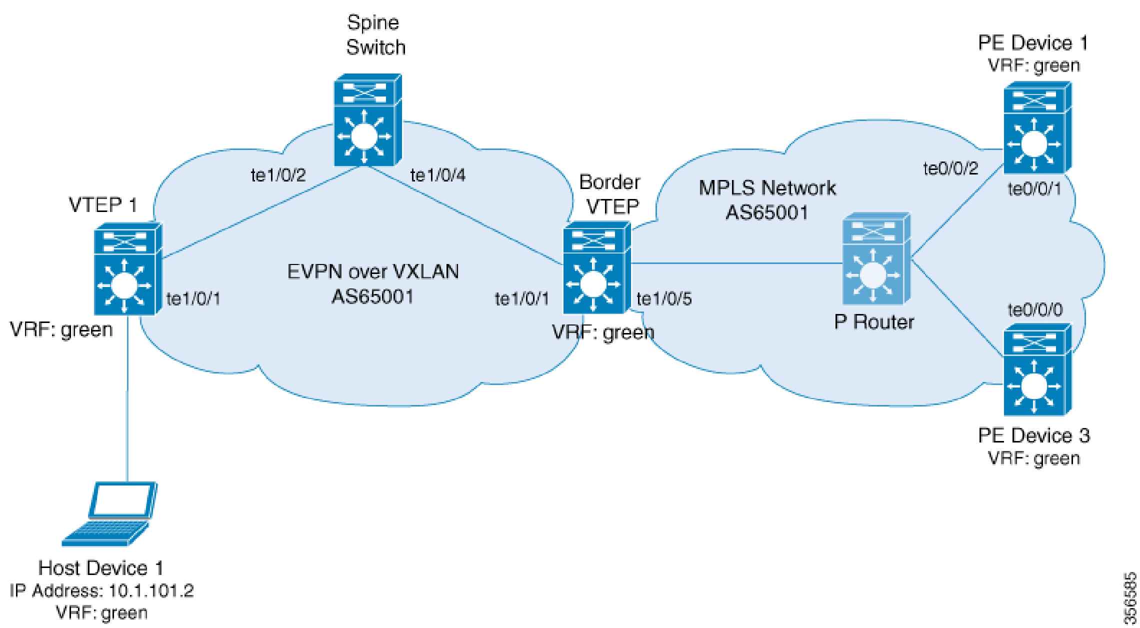
Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface GigabitEthernet1/0/30`	Enters the interface configuration mode for the specified interface.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding green`	Associates the VRF with the interface. **Note** The interface must be associated with the same VRF for which the Layer 3 VNI has been configured for the EVPN VXLAN network.
Step 5	**ip address** *ip-address* **Example:** `Device(config-if)# ip address 192.168.3.203 255.255.255.0`	Configures the IP address for the interface.
Step 6	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Enabling Layer 3 External Connectivity with MPLS Layer 3 VPN

The following figure shows a sample topology that illustrates Layer 3 external connectivity with an MPLS Layer 3 VPN network:



To enable EVPN VLAN Layer 3 external connectivity with MPLS Layer 3 VPN networks, perform the following set of procedures:

- Run the **mpls label mode all-vrfs protocol all-afs per-vrf** command in global configuration mode on the border VTEP.
- Configure BGP with reorigination of routes with a new route type for Layer 2 VPN, VPNv4, VPNv6 address families on the border VTEP.

Configuring BGP on a Border VTEP for External Connectivity with MPLS Layer 3 VPN

To configure BGP on a border VTEP to establish eternal connectivity with an MPLS Layer 3 VPN network, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 1`	Enables a BGP routing process, assigns it an autonomous system number, and enters router configuration mode.
Step 4	**bgp log-neighbor-changes** **Example:** `Device(config-router)# bgp log-neighbor-changes`	(Optional) Enables the generation of logging messages when the status of a BGP neighbor changes. For more information, see Configuring BGP section of the *IP Routing Configuration Guide*.
Step 5	**bgp update-delay** *time-period* **Example:** `Device(config-router)# bgp update-delay 1`	(Optional) Sets the maximum initial delay period before sending the first update. For more information, see Configuring BGP section of the *IP Routing Configuration Guide*.
Step 6	**bgp graceful-restart** **Example:** `Device(config-router)# bgp graceful-restart`	(Optional) Enables the BGP graceful restart capability for all BGP neighbors. For more information, see Configuring BGP section of the *IP Routing Configuration Guide*.

	Command or Action	Purpose
Step 7	**no bgp default ipv4-unicast** **Example:** `Device(config-router)# no bgp default ipv4-unicast`	(Optional) Disables default IPv4 unicast address family for BGP peering session establishment. For more information, see Configuring BGP section of the *IP Routing Configuration Guide*.
Step 8	**neighbor** *spine-ip-address* **remote-as** *number* **Example:** `Device(config-router)# neighbor 172.16.255.1 remote-as 1`	Defines multiprotocol-BGP neighbors in the EVPN network. Use the IP address of the spine switch as the neighbor IP address. This configures the spine switch as a BGP neighbor.
Step 9	**neighbor** *mpls-peer-ip-address* **remote-as** *number* **Example:** `Device(config-router)# neighbor 172.16.255.103 remote-as 1`	Defines multiprotocol-BGP neighbors in the external MPLS network. Use the IP address of the external MPLS network peer as the neighbor IP address. This configures the external MPLS network peer as a BGP neighbor.
Step 10	**neighbor** {*ip-address* | *group-name*} **update-source** *interface* **Example:** `Device(config-router)# neighbor 172.16.255.1 update-source Loopback0`	Configures update source. Update source can be configured per neighbor or per peer-group. Use the IP address of the spine switch as the neighbor IP address.
Step 11	**address-family l2vpn evpn** **Example:** `Device(config-router)# address-family l2vpn evpn`	Specifies the L2VPN address family and enters address family configuration mode.
Step 12	**import vpnv4 unicast re-originate** **Example:** `Device(config-router-af)# import vpnv4 unicast re-originate`	Reoriginates the VPNv4 routes imported from the external peer into the EVPN address family as EVPN routes, and distributes within the EVPN fabric.
Step 13	**import vpnv6 unicast re-originate** **Example:** `Device(config-router-af)# import vpnv6 unicast re-originate`	Reoriginates the VPNv6 routes imported from the external peer into the EVPN address family as EVPN routes, and distributes within the EVPN fabric.
Step 14	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 10.11.11.11 activate`	Enables the exchange information from a BGP neighbor. Use the IP address of the spine switch as the neighbor IP address.
Step 15	**neighbor** *ip-address* **send-community** [**both** | **extended** | **standard**]	Specifies the communities attribute sent to a BGP neighbor.

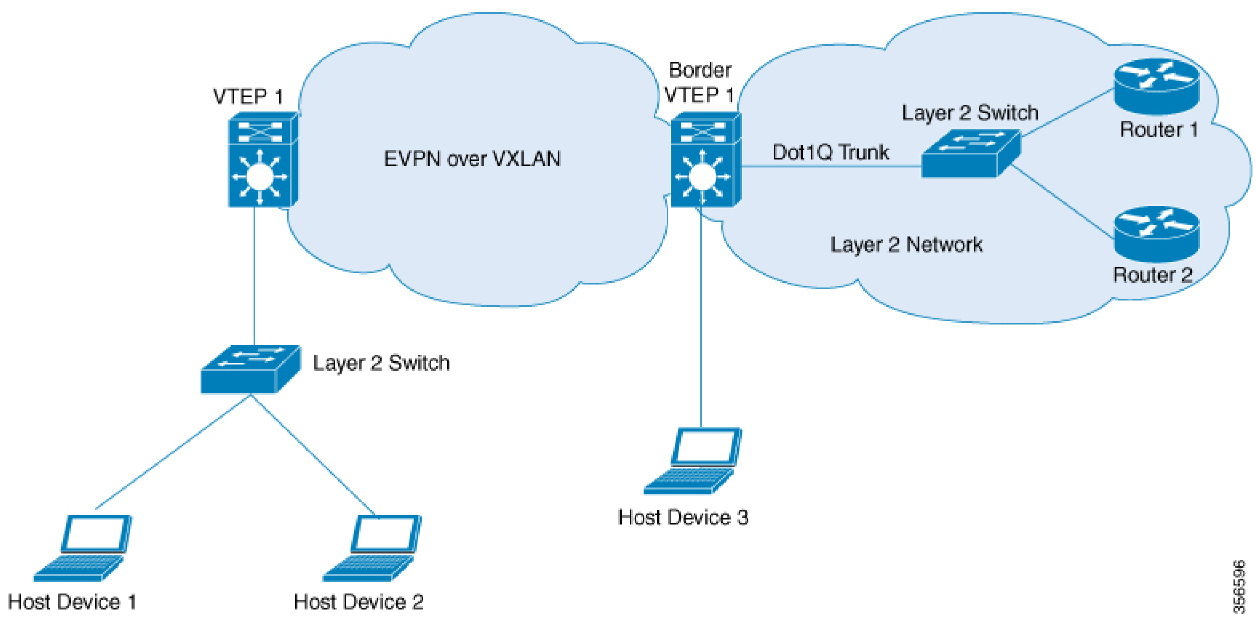
	Command or Action	Purpose
	Example: `Device(config-router-af)# neighbor 10.11.11.11 send-community both`	Use the IP address of the spine switch as the neighbor IP address. **Note** Use either of **extended** or **both** keywords. External connectivity cannot be established when you use the **standard** keyword.
Step 16	**neighbor** {**ip-address** | **peer-group-name**} **next-hop-self** [**all**] **Example:** `Device(config-router-af)# neighbor ip-address next-hop-self all`	Configures the router as the next hop for a BGP-speaking neighbor or peer group. The **all** keyword is mandatory when implementing external connectivity through iBGP, where the EVPN fabric and the MPLS network are in the same BGP autonomous system number. The **all** keyword is optional when implementing external connectivity through eBGP, where the EVPN fabric and the MPLS network are in different BGP autonomous system numbers.
Step 17	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 18	**address-family vpnv4** **Example:** `Device(config-router)# address-family vpnv4`	Specifies the VPNv4 address family and enters address family configuration mode.
Step 19	**import l2vpn evpn re-originate** **Example:** `Device(config-router-af)# import l2vpn evpn re-originate`	Reoriginates the EVPN routes imported from the EVPN fabric into the VPNv4 address family as VPNv4 routes and distributes them to the external network.
Step 20	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 172.16.255.103 activate`	Enables the exchange information from a BGP neighbor. Use the IP address of the external MPLS network router as the neighbor IP address.
Step 21	**neighbor** *ip-address* **send-community** [**both** | **extended** | **standard**] **Example:** `Device(config-router-af)# neighbor 172.16.255.103 send-community both`	Specifies the communities attribute sent to a BGP neighbor. Use the IP address of the external MPLS network router as the neighbor IP address.

	Command or Action	Purpose
		Note Use either of **extended** or **both** keywords. External connectivity cannot be established when you use the **standard** keyword.
Step 22	**neighbor** {**ip-address** | **peer-group-name**} **next-hop-self** [**all**] **Example:** `Device(config-router-af)# neighbor ip-address next-hop-self all`	Configures the router as the next hop for a BGP-speaking neighbor or peer group. The **all** keyword is mandatory when implementing external connectivity through iBGP, where the EVPN fabric and the MPLS network are in the same BGP autonomous system number. The **all** keyword is optional when implementing external connectivity through eBGP, where the EVPN fabric and the MPLS network are in different BGP autonomous system numbers.
Step 23	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 24	**address-family vpnv6** **Example:** `Device(config-router)# address-family vpnv6`	Specifies the VPNv6 address family and enters address family configuration mode.
Step 25	**import l2vpn evpn re-originate** **Example:** `Device(config-router-af)# import l2vpn evpn re-originate`	Reoriginates the EVPN routes imported from the EVPN fabric into the VPNv6 address family as VPNv6 routes and distributes them to the external network.
Step 26	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 172.16.255.103 activate`	Enables the exchange information from a BGP neighbor. Use the IP address of the spine switch as the neighbor IP address.
Step 27	**neighbor** *ip-address* **send-community** [**both** | **extended** | **standard**] **Example:** `Device(config-router-af)# neighbor 172.16.255.103 send-community both`	Specifies the communities attribute sent to a BGP neighbor. Use the IP address of the spine switch as the neighbor IP address. **Note** Use either of **extended** or **both** keywords. External connectivity cannot be established when you use the **standard** keyword.

	Command or Action	Purpose
Step 28	**neighbor** {**ip-address** | **peer-group-name**} **next-hop-self** [**all**] **Example:** `Device(config-router-af)# neighbor ip-address next-hop-self all`	Configures the router as the next hop for a BGP-speaking neighbor or peer group. The **all** keyword is mandatory when implementing external connectivity through iBGP, where the EVPN fabric and the MPLS network are in the same BGP autonomous system number. The **all** keyword is optional when implementing external connectivity through eBGP, where the EVPN fabric and the MPLS network are in different BGP autonomous system numbers.
Step 29	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 30	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.

Enabling Layer 2 External Connectivity with IEEE 802.1Q Networks

The following image shows a sample topology that illustrates Layer 2 external connectivity with an IEEE 802.1Q network:



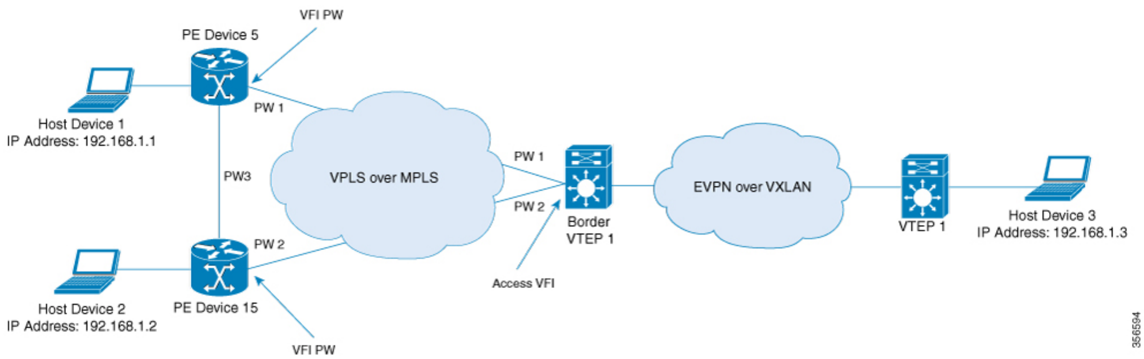
You can also connect the EVPN VXLAN network to a firewall in place of the Layer 2 switch in the above image. To configure Layer 2 external connectivity with an IEEE 802.1Q network, perform the following steps on the external Layer 2 switch:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface GigabitEthernet4/0/1`	Enters interface configuration mode for the specified interface. The specified interface must be the interface on the Layer 2 switch through which the EVPN VXLAN network communicates with the IEEE 802.1Q network.
Step 4	**switchport mode trunk** **Example:** `Device(config-if)# switchport mode trunk`	Configures the interface as a trunking VLAN Layer 2 interface.
Step 5	**switchport trunk allowed vlan** *vlan-list* **Example:** `Device(config-if)# switchport trunk allowed vlan 201,202`	Sets the list of VLANs that are allowed to transmit traffic from this interface in tagged format when the interface is in trunking mode.
Step 6	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Enabling Layer 2 External Connectivity with a VPLS Network Through an Access VFI

The following illustration shows a single-homed VXLAN network connected to a VPLS over MPLS network through the access VFIs on the border VTEP:



Note

We recommend you to use Cisco Catalyst 9500 Series - High Performance switches or Cisco Catalyst 9600 Series switches as border VTEPs when you configure Layer 2 external connectivity with a VPLS network.

We recommend you to configure Cisco Stackwise Virtual on the border VTEPs in order to achieve physical redundancy when you configure Layer 2 external connectivity with a VPLS network.

Perform the following set of procedures to enable Layer 2 external connectivity with VPLS networks through an access VFI interface:

1. Define the access VFI for the VTEPs.
2. Configure the access VFI as a member of the VLAN on the VTEPs.
3. Configure the EVPN instance as a member of the VLAN on the VTEPs.
4. Configure VPLS on the border VTEP.

Defining an Access VFI on a Border VTEP

To configure an access facing VFI on the VLAN of a border VTEP, perform the following steps:

For more information on configuring VFIs, in the software configuration guide for the required release, go to *Contents* → *Multiprotocol Label Switching (MPLS) Configuration Guide* → *Configuring Virtual Private LAN Service (VPLS) and VPLS BGP-Based Autodiscovery*.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enters privileged EXEC mode. Enter password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context myVFI`	Establishes an Layer 2 VPN VFI between two or more separate networks, and enters VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 1`	Configures the VPN ID for the VFI.
Step 5	**member** *ip-address* **encapsulation mpls** **Example:** `Device(config-vfi)# member 10.12.12.5 encapsulation mpls`	Specifies the device that forms a point-to-point Layer 2 VPN VFI connection.
Step 6	Repeat step 5 for all devices that form a point-to-point Layer 2 VPN VFI connection.	
Step 7	**end** **Example:** `Device(config-vfi)# end`	Exits VFI configuration mode and enters privileged EXEC mode.

Adding an Access VFI and an EVPN Instance as Members of the VLAN of a Border VTEP

To add an access VFI and an EVPN instance as members of the VLAN of a border VTEP, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enters privileged EXEC mode. Enter password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vlan configuration** *vlan-number* **Example:** `Device(config)# vlan configuration 11`	Enters VLAN feature configuration mode for the specified VLAN interface. Enter the VLAN number that is associated with the Layer 2 VNI configured in the EVPN VXLAN network.
Step 4	**member access-vfi** *vfi-name* **Example:**	Adds the access VFI as a member of the VLAN configuration.

	Command or Action	Purpose
	`Device(config-vlan)# member access-vfi myVFI`	
Step 5	**member evpn-instance** *evpn-instance-number* **vni** *l2-vni-number* **Example:** `Device(config-vlan)# member evpn-instance 1 vni 6000`	Adds the EVPN instance as a member of the VLAN configuration.
Step 6	**end** **Example:** `Device(config-vlan)# end`	Exits VLAN configuration mode and enters privileged EXEC mode.

Configuring VPLS on a Border VTEP

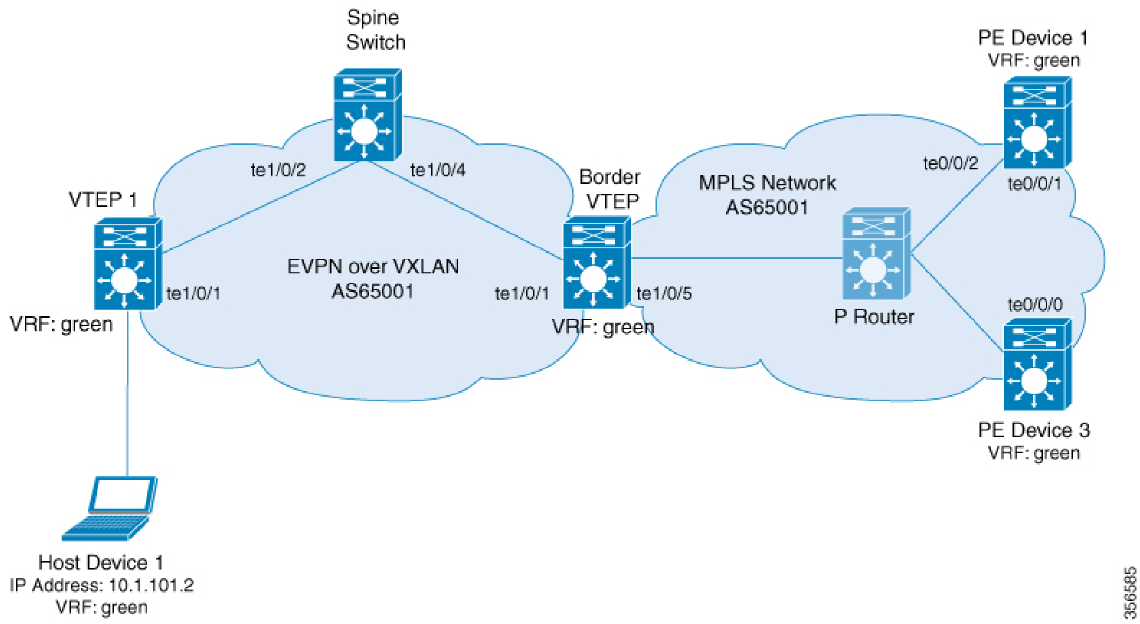
To configure VPLS on a border VTEP, in the software configuration guide for the required release, see *Contents* → *Multiprotocol Label Switching (MPLS) Configuration Guide* → *Configuring Virtual Private LAN Service (VPLS) and VPLS BGP-Based Autodiscovery*.

Configuration Examples for EVPN VXLAN External Connectivity

The following section shows the configuration examples for EVPN VXLAN external connectivity to other technologies:

Configuration Example for Enabling Layer 3 External Connectivity with MPLS Layer 3 VPN through iBGP

This section provides an example to show how Layer 3 external connectivity with MPLS Layer 3 VPN is enabled for a BGP EVPN VXLAN fabric through iBGP. The example shows how to configure and verify Layer 3 external connectivity with MPLS Layer 3 VPN for the topology shown below:



The topology shows an EVPN VXLAN network with two VTEPS, VTEP 1 and border VTEP. Border VTEP is connected to an external PE device that belongs to an MPLS network. The BGP EVPN VXLAN fabric and the MPLS network are in the autonomous system number 65001. All the VTEPs, PE devices and, host devices are part of the VRF green. The following tables provide sample configurations for the devices in the topology above.

Table 8: Configuring Spine Switch, Border VTEP and PE Device 1 for Enabling Layer 3 External Connectivity with MPLS Layer 3 VPN through iBGP

Spine Switch	Border VTEP	PE Device 1
Spine_switch# **show running-config** hostname Spine_switch ! interface Loopback0 ip address 172.16.255.1 255.255.255.255 ip ospf 1 area 0 ip pim sparse-mode ! interface Loopback1 ip address 172.16.254.1 255.255.255.255 ip pim sparse-mode ip ospf 1 area 0 ! interface Loopback2 ip address 172.16.255.255 255.255.255.255 ip pim sparse-mode ip ospf 1 area 0 ! interface TenGigabitEthernet1/0/2 no switchport ip address 172.16.14.1 255.255.255.0 ip pim sparse-mode ip ospf network point-to-point ip ospf 1 area 0 ! interface TenGigabitEthernet1/0/4 no switchport ip address 172.16.16.1 255.255.255.0 ip pim sparse-mode ip ospf network point-to-point ip ospf 1 area 0 ! router ospf 1 router-id 172.16.255.1 ! router bgp 65001 template peer-policy RR-PP route-reflector-client send-community both exit-peer-policy ! template peer-session RR-PS remote-as 65001 update-source Loopback0 exit-peer-session ! bgp router-id 172.16.255.1 bgp log-neighbor-changes no bgp default ipv4-unicast neighbor 172.16.255.4 inherit peer-session RR-PS neighbor 172.16.255.6 inherit peer-session RR-PS ! ! !	Border_VTEP# **show running-config** hostname Border_VTEP !vrf definition green rd 1:1 ! address-family ipv4 route-target export 1:1 route-target import 1:1 route-target export 1:1 stitching route-target import 1:1 stitching exit-address-family ! address-family ipv6 route-target export 1:1 route-target import 1:1 route-target export 1:1 stitching route-target import 1:1 stitching exit-address-family ! mpls label mode all-vrfs protocol all-afs per-vrf ! l2vpn evpn replication-type static router-id Loopback1 default-gateway advertise ! l2vpn evpn instance 101 vlan-based encapsulation vxlan ! l2vpn evpn instance 102 vlan-based encapsulation vxlan replication-type ingress ! vlan configuration 101 member evpn-instance 101 vni 10101 vlan configuration 102 member evpn-instance 102 vni 10102 vlan configuration 901 member vni 50901 ! interface Loopback0 ip address 172.16.255.6 255.255.255.255 ip pim sparse-mode ip ospf 1 area 0 ! interface Loopback1 ip address 172.16.254.6 255.255.255.255 ip pim sparse-mode ip ospf 1 area 0 ! interface TenGigabitEthernet1/0/1 no switchport ip address 172.16.16.6 255.255.255.0 ip pim sparse-mode ip ospf network point-to-point ip ospf 1 area 0	PE_device_1# **show running-config** hostname PE_device_1 ! vrf definition green rd 1:1 ! address-family ipv4 route-target export 1:1 route-target import 1:1 exit-address-family ! address-family ipv6 route-target export 1:1 route-target import 1:1 exit-address-family ! interface Loopback0 ip address 172.16.255.101 255.255.255.255 ! interface Loopback1 vrf forwarding green ip address 10.1.255.101 255.255.255.255 ! interface TenGigabitEthernet0/0/1 ip address 172.16.111.101 255.255.255.0 ip router isis cdp enable mpls ip isis network point-to-point ! interface TenGigabitEthernet0/0/2 ip address 172.16.106.101 255.255.255.0 ip router isis negotiation auto cdp enable mpls ip isis network point-to-point ! router isis net 49.0001.1720.1625.5101.00 is-type level-2-only metric-style wide passive-interface Loopback0 ! router bgp 65001 bgp log-neighbor-changes no bgp default ipv4-unicast neighbor 172.16.255.103 remote-as 65001 neighbor 172.16.255.103 update-source Loopback0 ! address-family ipv4 exit-address-family !

Spine Switch	Border VTEP	PE Device 1
! address-family ipv4 exit-address-family ! address-family l2vpn evpn neighbor 172.16.255.4 activate neighbor 172.16.255.4 send-community extended neighbor 172.16.255.4 inherit peer-policy RR-PP neighbor 172.16.255.6 activate neighbor 172.16.255.6 send-community extended neighbor 172.16.255.6 inherit peer-policy RR-PP exit-address-family ! ip pim rp-address 172.16.255.255 ! end !	! interface TenGigabitEthernet1/0/5 no switchport ip address 172.16.106.6 255.255.255.0 ip router isis duplex full mpls ip isis network point-to-point ! interface Vlan101 vrf forwarding green ip address 10.1.101.1 255.255.255.0 ! interface Vlan102 vrf forwarding green ip address 10.1.102.1 255.255.255.0 ! interface Vlan901 vrf forwarding green ip unnumbered Loopback1 ipv6 enable no autostate ! interface nve1 no ip address source-interface Loopback1 host-reachability protocol bgp member vni 10101 mcast-group 225.0.0.101 member vni 50901 vrf green member vni 10102 ingress-replication ! router ospf 1 ! router isis net 49.0001.1720.1625.5006.00 is-type level-2-only metric-style wide passive-interface Loopback0 ! router bgp 65001 ! template peer-session RR-PS remote-as 65001 update-source Loopback0 exit-peer-session ! bgp log-neighbor-changes no bgp default ipv4-unicast neighbor 172.16.255.1 inherit peer-session RR-PS neighbor 172.16.255.103 inherit peer-session RR-PS ! address-family ipv4 exit-address-family ! ! ! ! !	address-family vpnv4 neighbor 172.16.255.103 activate neighbor 172.16.255.103 send-community both exit-address-family ! address-family vpnv6 neighbor 172.16.255.103 activate neighbor 172.16.255.103 send-community both exit-address-family ! address-family ipv4 vrf green redistribute connected exit-address-family ! address-family ipv6 vrf green redistribute connected exit-address-family ! end !

Spine Switch	Border VTEP	PE Device 1
!	!	!
!	address-family vpnv4	!
!	import l2vpn evpn re-originate	!
!	neighbor 172.16.255.103 activate	!
!	neighbor 172.16.255.103	!
!	send-community both	!
!	neighbor 172.16.255.103	!
!	route-reflector-client	!
!	neighbor 172.16.255.103 next-hop-self	!
!	all	!
!	exit-address-family	!
!	!	!
!	address-family vpnv6	!
!	import l2vpn evpn re-originate	!
!	neighbor 172.16.255.103 activate	!
!	neighbor 172.16.255.103	!
!	send-community both	!
!	neighbor 172.16.255.103	!
!	route-reflector-client	!
!	neighbor 172.16.255.103 next-hop-self	!
!	all	!
!	exit-address-family	!
!	!	!
!	address-family l2vpn evpn	!
!	import vpnv4 unicast re-originate	!
!	import vpnv6 unicast re-originate	!
!	neighbor 172.16.255.1 activate	!
!	neighbor 172.16.255.1 send-community	!
!	both	!
!	neighbor 172.16.255.1	!
!	route-reflector-client	!
!	neighbor 172.16.255.1 next-hop-self	!
!	all	!
!	exit-address-family	!
!	!	!
!	address-family ipv4 vrf green	!
!	advertise l2vpn evpn	!
!	redistribute connected	!
!	redistribute static	!
!	exit-address-family	!
!	!	!
!	address-family ipv6 vrf green	!
!	advertise l2vpn evpn	!
!	redistribute connected	!
!	redistribute static	!
!	exit-address-family	!
!	!	!
!	ip pim rp-address 172.16.255.255	!
!	!	!
!	end	!
Spine_switch#	Border_VTEP#	PE_device_1#

Table 9: Configuring VTEP 1 and PE Device 3 for Enabling Layer 3 External Connectivity with MPLS Layer 3 VPN through iBGP

VTEP 1	PE Device 3
VTEP_1# **show running-config** hostname VTEP_1 ! vrf definition green rd 1:1 ! address-family ipv4 route-target export 1:1 route-target import 1:1 route-target export 1:1 stitching route-target import 1:1 stitching exit-address-family ! address-family ipv6 route-target export 1:1 route-target import 1:1 route-target export 1:1 stitching route-target import 1:1 stitching exit-address-family ! l2vpn evpn replication-type static router-id Loopback1 default-gateway advertise ! l2vpn evpn instance 101 vlan-based encapsulation vxlan ! l2vpn evpn instance 102 vlan-based encapsulation vxlan replication-type ingress ! vlan configuration 101 member evpn-instance 101 vni 10101 vlan configuration 102 member evpn-instance 102 vni 10102 vlan configuration 901 member vni 50901 ! interface Loopback0 ip address 172.16.255.4 255.255.255.255 ip pim sparse-mode ip ospf 1 area 0 ! interface Loopback1 ip address 172.16.254.4 255.255.255.255 ip pim sparse-mode ip ospf 1 area 0 ! interface TenGigabitEthernet1/0/1 no switchport ip address 172.16.14.4 255.255.255.0 ip pim sparse-mode ip ospf network point-to-point ip ospf 1 area 0 ! interface Vlan101 vrf forwarding green ip address 10.1.101.1 255.255.255.0	PE_device_3# **show running-config** hostname PE_device_3 ! vrf definition green rd 1:1 ! address-family ipv4 route-target export 1:1 route-target import 1:1 exit-address-family ! address-family ipv6 route-target export 1:1 route-target import 1:1 exit-address-family ! interface Loopback0 ip address 172.16.255.103 255.255.255.255 ! interface Loopback1 vrf forwarding green ip address 10.1.255.103 255.255.255.255 ! interface TenGigabitEthernet0/0/0 ip address 172.16.111.103 255.255.255.0 ip router isis cdp enable mpls ip isis network point-to-point ! router isis net 49.0001.1720.1625.5103.00 is-type level-2-only metric-style wide passive-interface Loopback0 ! router bgp 65001 template peer-policy RR-PP route-reflector-client send-community both exit-peer-policy ! template peer-session RR-PS remote-as 65001 update-source Loopback0 exit-peer-session ! bgp log-neighbor-changes no bgp default ipv4-unicast neighbor 172.16.255.6 inherit peer-session RR-PS neighbor 172.16.255.101 inherit peer-session RR-PS ! address-family ipv4 exit-address-family ! ! ! ! !

<table>
<tr><th>VTEP 1</th><th>PE Device 3</th></tr>
<tr><td><pre>
!
interface Vlan102
vrf forwarding green
ip address 10.1.102.1 255.255.255.0
!
interface Vlan901
vrf forwarding green
ip unnumbered Loopback1
ipv6 enable
no autostate
!
interface nve1
no ip address
source-interface Loopback1
host-reachability protocol bgp
member vni 10101 mcast-group 225.0.0.101
member vni 50901 vrf green
member vni 10102 ingress-replication
!
router ospf 1
!
router bgp 65001
!
template peer-session RR-PS
remote-as 65001
update-source Loopback0
exit-peer-session
!
bgp log-neighbor-changes
no bgp default ipv4-unicast
neighbor 172.16.255.1 inherit peer-session RR-PS
!
address-family ipv4
exit-address-family
!
address-family l2vpn evpn
neighbor 172.16.255.1 activate
neighbor 172.16.255.1 send-community both
exit-address-family
!
address-family ipv4 vrf green
advertise l2vpn evpn
redistribute connected
redistribute static
exit-address-family
!
address-family ipv6 vrf green
advertise l2vpn evpn
redistribute connected
redistribute static
exit-address-family
!
ip pim rp-address 172.16.255.255
!
end
!
!

VTEP_1#
</pre></td><td><pre>
address-family vpnv4
neighbor 172.16.255.6 activate
neighbor 172.16.255.6 send-community extended
neighbor 172.16.255.6 inherit peer-policy RR-PP
neighbor 172.16.255.101 activate
neighbor 172.16.255.101 send-community extended
neighbor 172.16.255.101 inherit peer-policy RR-PP
exit-address-family
!
address-family vpnv6
neighbor 172.16.255.6 activate
neighbor 172.16.255.6 send-community extended
neighbor 172.16.255.6 inherit peer-policy RR-PP
neighbor 172.16.255.101 activate
neighbor 172.16.255.101 send-community extended
neighbor 172.16.255.101 inherit peer-policy RR-PP
exit-address-family
!
address-family ipv4 vrf green
redistribute connected
exit-address-family
!
address-family ipv6 vrf green
redistribute connected
exit-address-family
!
end
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!

PE_device_3#
</pre></td></tr>
</table>

The following examples provide sample outputs for **show** commands on VTEP 1 and border VTEP to verify external connectivity for the topology configured above:

VTEP 1

The following example shows the output for the **show bgp l2vpn evpn route-type** command for route type 5 on VTEP 1:

```
VTEP_1# show bgp l2vpn evpn route-type 5 0 10.1.255.103 32
BGP routing table entry for [5][1:1][0][32][10.1.255.103]/17, version 12
Paths: (1 available, best #1, table EVPN-BGP-Table)
  Flag: 0x100
  Not advertised to any peer
  Refresh Epoch 1
  Local
    172.16.254.6 (metric 3) (via default) from 172.16.255.1 (172.16.255.1)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Gateway Address: 0.0.0.0, VNI Label 50901, MPLS VPN
Label 0
      Extended Community: RT:1:1 ENCAP:8 Router MAC:0C75.BD67.EF48
      Originator: 172.16.255.103, Cluster list: 172.16.255.1, 172.16.255.6
      rx pathid: 0, tx pathid: 0x0
      net: 0x7F84B914EF38, path: 0x7F84BAFD0E30, pathext: 0x7F84BB42E698
      flags: net: 0x100, path: 0x3, pathext: 0xA1
      Updated on May 20 2020 19:31:08 UTC
```

The following example shows the output for the **show bgp l2vpn evpn route-type** command for route type 2 on VTEP 1:

```
VTEP_1# show bgp l2vpn evpn route-type 2 0 44d3ca286cc1 10.1.101.2
BGP routing table entry for [2][172.16.254.4:101][0][48][44D3CA286CC1][32][10.1.101.2]/24,
 version 17
Paths: (1 available, best #1, table evi_101)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  Local
    :: (via default) from 0.0.0.0 (172.16.255.4)
      Origin incomplete, localpref 100, weight 32768, valid, sourced, local, best
      EVPN ESI: 00000000000000000000, Label1 10101, Label2 50901
      Extended Community: RT:1:1 RT:65001:101 ENCAP:8
        Router MAC:7C21.0DBD.9548
      Local irb vxlan vtep:
        vrf:green, l3-vni:50901
        local router mac:7C21.0DBD.9548
        core-irb interface:Vlan901
        vtep-ip:172.16.254.4
      rx pathid: 0, tx pathid: 0x0
      net: 0x7F84B914E858, path: 0x7F84BAFD09F8, pathext: 0x7F84BB42E4B8
      flags: net: 0x0, path: 0x4000028000003, pathext: 0x81
      Updated on May 20 2020 19:31:30 UTC
```

The following example shows the output for the **show ip route vrf** command on VTEP 1:

```
VTEP_1# show ip route vrf green

Routing Table: green
```

```
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
C        10.1.101.0/24 is directly connected, Vlan101
L        10.1.101.1/32 is directly connected, Vlan101
C        10.1.102.0/24 is directly connected, Vlan102
L        10.1.102.1/32 is directly connected, Vlan102
B        10.1.255.101/32 [200/0] via 172.16.254.6, 00:21:47, Vlan901
B        10.1.255.103/32 [200/0] via 172.16.254.6, 00:21:47, Vlan901
```

Border VTEP

The following example shows the output for the **show mpls ldp neighbor** command on border VTEP:

```
Border_VTEP# show mpls ldp neighbor
    Peer LDP Ident: 172.16.111.101:0; Local LDP Ident 172.16.106.6:0
        TCP connection: 172.16.111.101.26371 - 172.16.106.6.646
        State: Oper; Msgs sent/rcvd: 86/69; Downstream
        Up time: 00:32:14
        LDP discovery sources:
          TenGigabitEthernet1/0/5, Src IP addr: 172.16.106.101
        Addresses bound to peer LDP Ident:
          172.16.111.101  172.16.106.101  172.16.255.101
```

The following example shows the output for the **show bgp l2vpn evpn route-type** command for route type 5 on border VTEP:

```
Border_VTEP# show bgp l2vpn evpn route-type 5 0 10.1.255.103 32
BGP routing table entry for [5][1:1][0][32][10.1.255.103]/17, version 7
Paths: (1 available, best #1, table EVPN-BGP-Table)
  Flag: 0x100
  Advertised to update-groups:
     1
  Refresh Epoch 1
  Local, (Received from a RR-client), imported path from base
    172.16.255.103 (metric 20) (via default) from 172.16.255.103 (172.16.255.103)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Gateway Address: 0.0.0.0, local vtep: 172.16.254.6,
VNI Label 50901, MPLS VPN Label 23
      Extended Community: RT:1:1 ENCAP:8 Router MAC:0C75.BD67.EF48
      rx pathid: 0, tx pathid: 0x0
      net: 0x7FED6F808948, path: 0x7FED6D7EDA68, pathext: 0x7FED6D80DE40, exp_net:
0x7FED6F9BF070
      flags: net: 0x100, path: 0x7, pathext: 0xA1
      Updated on May 20 2020 19:22:47 UTC
```

The following example shows the output for the **show bgp vpnv4 unicast all** command on border VTEP for the IP address of host device 1:

```
Border_VTEP# show bgp vpnv4 unicast all 10.1.101.2
BGP routing table entry for 1:1:10.1.101.2/32, version 10
Paths: (1 available, best #1, table green)
  Advertised to update-groups:
     3
  Refresh Epoch 1
  Local, (Received from a RR-client), imported path from
[2][172.16.254.4:101][0][48][44D3CA286CC1][32][10.1.101.2]/24 (global)
    172.16.254.4 (metric 3) (via default) from 172.16.255.1 (172.16.255.1)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:1:1 ENCAP:8 Router MAC:7C21.0DBD.9548
      Originator: 172.16.255.4, Cluster list: 172.16.255.1
      Local vxlan vtep:
        vrf:green, vni:50901
        local router mac:0C75.BD67.EF48
        encap:8
        vtep-ip:172.16.254.6
        bdi:Vlan901
      Remote VxLAN:
        Topoid 0x4(vrf green)
        Remote Router MAC:7C21.0DBD.9548
        Encap 8
        Egress VNI 50901
        RTEP 172.16.254.4
      mpls labels in/out IPv4 VRF Aggr:34/nolabel
      rx pathid: 0, tx pathid: 0x0
      Updated on May 20 2020 19:23:11 UTC
```

Spine Switch

The following example shows the output for the **show bgp l2vpn evpn route-type** command for route type 5 on spine switch:

```
Spine_switch# show bgp l2vpn evpn route-type 5 0 10.1.255.103 32
BGP routing table entry for [5][1:1][0][32][10.1.255.103]/17, version 12
Paths: (1 available, best #1, table EVPN-BGP-Table)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  Local, (Received from a RR-client)
    172.16.254.6 (metric 2) (via default) from 172.16.255.6 (172.16.255.6)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Gateway Address: 0.0.0.0, VNI Label 50901, MPLS VPN
Label 0
      Extended Community: RT:1:1 ENCAP:8 Router MAC:0C75.BD67.EF48
      Originator: 172.16.255.103, Cluster list: 172.16.255.6
      rx pathid: 0, tx pathid: 0x0
      net: 0x7F54CC99CEF8, path: 0x7F54CC9AD310, pathext: 0x7F54CC9C6998
      flags: net: 0x0, path: 0x3, pathext: 0x81
      Updated on May 20 2020 19:28:59 UTC
```

The following example shows the output for the **show bgp l2vpn evpn route-type** command for route type 2 on spine switch:

```
Spine_switch# show bgp l2vpn evpn route-type 2 0 44d3ca286cc1 10.1.101.2
BGP routing table entry for [2][172.16.254.4:101][0][48][44D3CA286CC1][32][10.1.101.2]/24,
 version 14
Paths: (1 available, best #1, table EVPN-BGP-Table)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  Local, (Received from a RR-client)
    172.16.254.4 (metric 2) (via default) from 172.16.255.4 (172.16.255.4)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Label1 10101, Label2 50901
      Extended Community: RT:1:1 RT:65001:101 ENCAP:8
        Router MAC:7C21.0DBD.9548
      rx pathid: 0, tx pathid: 0x0
      net: 0x7F54CC99CAD8, path: 0x7F54CC9AD088, pathext: 0x7F54CC9C68D8
      flags: net: 0x0, path: 0x3, pathext: 0x81
      Updated on May 20 2020 19:29:22 UTC
```

PE Device 3

The following example shows the output for the **show bgp vpnv4 unicast all** command on PE device 3 for the IP address of host device 1:

```
PE_device_3# show bgp vpnv4 unicast all 10.1.101.2
BGP routing table entry for 1:1:10.1.101.2/32, version 14
Paths: (1 available, best #1, table green)
  Advertised to update-groups:
     3
  Refresh Epoch 1
  Local, (Received from a RR-client)
    172.16.255.6 (metric 20) (via default) from 172.16.255.6 (172.16.255.6)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:1:1 ENCAP:8 Router MAC:7C21.0DBD.9548
      Originator: 172.16.255.4, Cluster list: 172.16.255.6, 172.16.255.1
      mpls labels in/out nolabel/34
      rx pathid: 0, tx pathid: 0x0
      Updated on May 20 2020 11:27:25 UTC
```

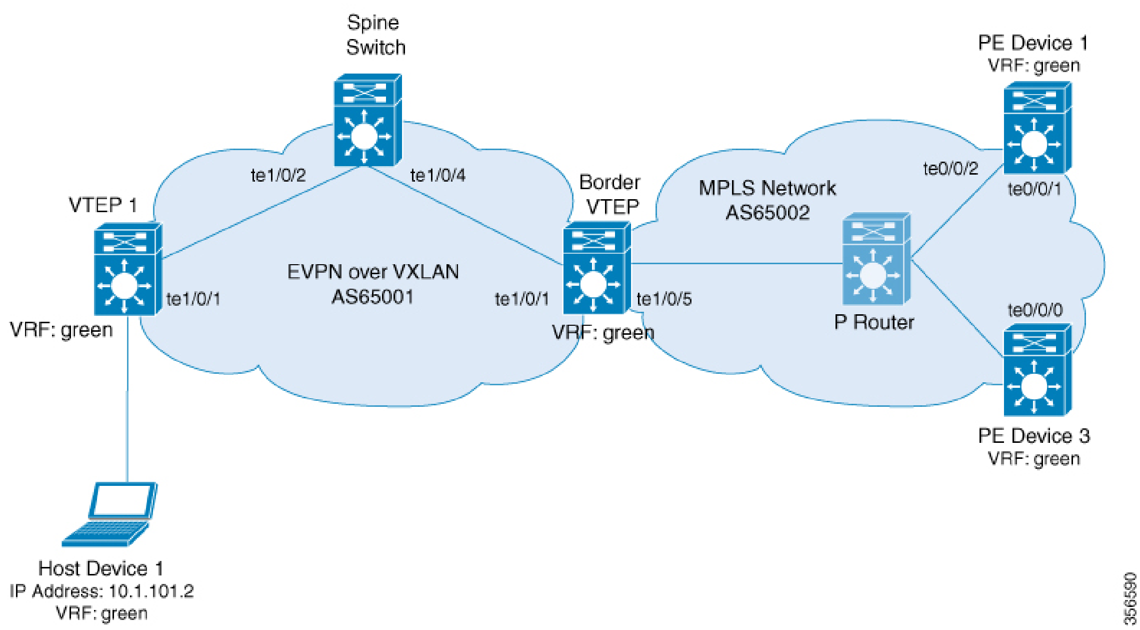
The following example shows the output for the **show ip route vrf green** command on PE device 3:

```
PE_device_3# show ip route vrf green

Routing Table: green
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
```

```
B        10.1.101.0/24 [200/0] via 172.16.255.6, 00:28:12
B        10.1.101.1/32 [200/0] via 172.16.255.6, 00:28:10
B        10.1.101.2/32 [200/0] via 172.16.255.6, 00:27:48
B        10.1.102.0/24 [200/0] via 172.16.255.6, 00:28:12
B        10.1.102.1/32 [200/0] via 172.16.255.6, 00:28:10
B        10.1.255.101/32 [200/0] via 172.16.255.101, 00:28:09
C        10.1.255.103/32 is directly connected, Loopback1
```

Configuration Example for Enabling Layer 3 External Connectivity with MPLS Layer 3 VPN through eBGP

This section provides an example to show how Layer 3 external connectivity with MPLS Layer 3 VPN is enabled for a BGP EVPN VXLAN fabric through eBGP. The example shows how to configure and verify Layer 3 external connectivity with MPLS Layer 3 VPN for the topology shown below:



The topology shows an EVPN VXLAN network with two VTEPS, VTEP 1 and border VTEP. Border VTEP is connected to an external PE device that belongs to an MPLS network. The BGP EVPN VXLAN fabric is in the autonomous system number 65001. The MPLS network is in the autonomous system number 65002. All the VTEPs, PE devices, and host devices are part of the VRF green. The following tables provide sample configurations for the devices in the topology above.

Table 10: Configuring Spine Switch, Border VTEP and PE Device 1 for Enabling Layer 3 External Connectivity with MPLS Layer 3 VPN through eBGP

Spine Switch	Border VTEP	PE Device 1
Spine_switch# **show running-config**	Border_VTEP# **show running-config**	PE_device_1# **show running-config**
hostname Spine_switch	hostname Border_VTEP	hostname PE_device_1
!	!	!
interface Loopback0	vrf definition green	vrf definition green
ip address 172.16.255.1	rd 1:1	rd 1:1
255.255.255.255	!	!
ip ospf 1 area 0	address-family ipv4	address-family ipv4
ip pim sparse-mode	route-target export 1:1	route-target export 1:1
!	route-target import 1:1	route-target import 1:1
interface Loopback1	route-target export 1:1 stitching	exit-address-family
ip address 172.16.254.1	route-target import 1:1 stitching	!
255.255.255.255	exit-address-family	address-family ipv6
ip pim sparse-mode	!	route-target export 1:1
ip ospf 1 area 0	address-family ipv6	route-target import 1:1
!	route-target export 1:1	exit-address-family
interface Loopback2	route-target import 1:1	!
ip address 172.16.255.255	route-target export 1:1 stitching	interface Loopback0
255.255.255.255	route-target import 1:1 stitching	ip address 172.16.255.101
ip pim sparse-mode	exit-address-family	255.255.255.255
ip ospf 1 area 0	!	!
!	mpls label mode all-vrfs protocol	interface Loopback1
interface TenGigabitEthernet1/0/2	all-afs per-vrf	vrf forwarding green
no switchport	!	ip address 10.1.255.101
ip address 172.16.14.1 255.255.255.0	l2vpn evpn	255.255.255.255
ip pim sparse-mode	replication-type static	!
ip ospf network point-to-point	router-id Loopback1	interface TenGigabitEthernet0/0/1
ip ospf 1 area 0	default-gateway advertise	ip address 172.16.111.101
!	!	255.255.255.0
interface TenGigabitEthernet1/0/4	l2vpn evpn instance 101 vlan-based	ip router isis
no switchport	encapsulation vxlan	cdp enable
ip address 172.16.16.1 255.255.255.0	!	mpls ip
ip pim sparse-mode	l2vpn evpn instance 102 vlan-based	isis network point-to-point
ip ospf network point-to-point	encapsulation vxlan	!
ip ospf 1 area 0	replication-type ingress	interface TenGigabitEthernet0/0/2
!	!	ip address 172.16.106.101
router ospf 1	vlan configuration 101	255.255.255.0
router-id 172.16.255.1	member evpn-instance 101 vni 10101	negotiation auto
!	vlan configuration 102	cdp enable
router bgp 65001	member evpn-instance 102 vni 10102	mpls bgp forwarding
template peer-policy RR-PP	vlan configuration 901	!
route-reflector-client	member vni 50901	router isis
send-community both	!	net 49.0001.1720.1625.5101.00
exit-peer-policy	interface Loopback0	is-type level-2-only
!	ip address 172.16.255.6	metric-style wide
template peer-session RR-PS	255.255.255.255	passive-interface Loopback0
remote-as 65001	ip pim sparse-mode	!
update-source Loopback0	ip ospf 1 area 0	router bgp 65002
exit-peer-session	!	bgp log-neighbor-changes
!	interface Loopback1	no bgp default ipv4-unicas
bgp router-id 172.16.255.1	ip address 172.16.254.6	no bgp default route-target filter
bgp log-neighbor-changes	255.255.255.255	neighbor 172.16.106.6 remote-as 65001
no bgp default ipv4-unicast	ip pim sparse-mode	neighbor 172.16.255.6 remote-as 65001
neighbor 172.16.255.4 inherit	ip ospf 1 area 0	neighbor 172.16.255.6 ebgp-multihop
peer-session RR-PS	!	255
neighbor 172.16.255.6 inherit	interface TenGigabitEthernet1/0/1	neighbor 172.16.255.6 update-source
peer-session RR-PS	no switchport	Loopback0
!	ip address 172.16.16.6 255.255.255.0	neighbor 172.16.255.103 remote-as
address-family ipv4	ip pim sparse-mode	65002
exit-address-family	ip ospf network point-to-point	neighbor 172.16.255.103 update-source
!	ip ospf 1 area 0	Loopback0

Spine Switch	**Border VTEP**	**PE Device 1**
! address-family l2vpn evpn neighbor 172.16.255.4 activate neighbor 172.16.255.4 send-community extended neighbor 172.16.255.4 inherit peer-policy RR-PP neighbor 172.16.255.6 activate neighbor 172.16.255.6 send-community extended neighbor 172.16.255.6 inherit peer-policy RR-PP exit-address-family ! ip pim rp-address 172.16.255.255 ! end !	! interface TenGigabitEthernet1/0/5 no switchport ip address 172.16.106.6 255.255.255.0 speed 1000 duplex full mpls bgp forwarding ! interface Vlan101 vrf forwarding green ip address 10.1.101.1 255.255.255.0 ! interface Vlan102 vrf forwarding green ip address 10.1.102.1 255.255.255.0 ! interface Vlan901 vrf forwarding green ip unnumbered Loopback1 ipv6 enable no autostate ! interface nve1 no ip address source-interface Loopback1 host-reachability protocol bgp member vni 10101 mcast-group 225.0.0.101 member vni 50901 vrf green member vni 10102 ingress-replication ! router ospf 1 ! router bgp 65001 template peer-policy RR-PP send-community both exit-peer-policy ! template peer-session RR-PS remote-as 65001 update-source Loopback0 exit-peer-session ! bgp log-neighbor-changes no bgp default ipv4-unicast no bgp default route-target filter neighbor 172.16.106.101 remote-as 65002 neighbor 172.16.255.1 inherit peer-session RR-PS neighbor 172.16.255.101 remote-as 65002 neighbor 172.16.255.101 ebgp-multihop 255 neighbor 172.16.255.101 update-source Loopback0 !	address-family ipv4 network 172.16.255.101 mask 255.255.255.255 neighbor 172.16.106.6 activate neighbor 172.16.106.6 send-label exit-address-family ! address-family vpnv4 neighbor 172.16.255.6 activate neighbor 172.16.255.6 send-community both neighbor 172.16.255.103 activate neighbor 172.16.255.103 send-community both neighbor 172.16.255.103 next-hop-self exit-address-family ! address-family vpnv6 neighbor 172.16.255.6 activate neighbor 172.16.255.6 send-community both neighbor 172.16.255.103 activate neighbor 172.16.255.103 send-community both neighbor 172.16.255.103 next-hop-self exit-address-family ! address-family ipv4 vrf green redistribute connected exit-address-family ! address-family ipv6 vrf green redistribute connected exit-address-family ! end !

Spine Switch	Border VTEP	PE Device 1
! ! ! !! Spine_switch#	 address-family ipv4 network 172.16.255.6 mask 255.255.255.255 neighbor 172.16.106.101 activate neighbor 172.16.106.101 send-label exit-address-family ! address-family vpnv4 import l2vpn evpn re-originate neighbor 172.16.255.101 activate neighbor 172.16.255.101 send-community both exit-address-family ! address-family vpnv6 import l2vpn evpn re-originate neighbor 172.16.255.101 activate neighbor 172.16.255.101 send-community both exit-address-family ! address-family l2vpn evpn import vpnv4 unicast re-originate import vpnv6 unicast re-originate neighbor 172.16.255.1 activate neighbor 172.16.255.1 send-community both neighbor 172.16.255.1 next-hop-self exit-address-family ! address-family ipv4 vrf green advertise l2vpn redistribute connected redistribute static exit-address-family ! address-family ipv6 vrf green advertise l2vpn evpn redistribute connected redistribute static exit-address-family ! ip pim rp-address 172.16.255.255 ! end Border_VTEP#	! PE_device_1#

Table 11: Configuring VTEP 1 and PE Device 3 for Enabling Layer 3 External Connectivity with MPLS Layer 3 VPN through eBGP

VTEP 1	PE Device 3

`VTEP_1#` **`show running-config`**
`hostname VTEP_1!`
`!`
`vrf definition green`
`rd 1:1`
`!`
`address-family ipv4`
`route-target export 1:1`
`route-target import 1:1`
`route-target export 1:1 stitching`
`route-target import 1:1 stitching`
`exit-address-family`
`!`
`address-family ipv6`
`route-target export 1:1`
`route-target import 1:1`
`route-target export 1:1 stitching`
`route-target import 1:1 stitching`
`exit-address-family`
`!`
`l2vpn evpn`
`replication-type static`
`router-id Loopback1`
`default-gateway advertise`
`!`
`l2vpn evpn instance 101 vlan-based`
`encapsulation vxlan`
`!`
`l2vpn evpn instance 102 vlan-based`
`encapsulation vxlan`
`replication-type ingress`
`!`
`vlan configuration 101`
`member evpn-instance 101 vni 10101`
`vlan configuration 102`
`member evpn-instance 102 vni 10102`
`vlan configuration 901`
`member vni 50901`
`!`
`interface Loopback0`
`ip address 172.16.255.4 255.255.255.255`
`ip pim sparse-mode`
`ip ospf 1 area 0`
`!`
`interface Loopback1`
`ip address 172.16.254.4 255.255.255.255`
`ip pim sparse-mode`
`ip ospf 1 area 0`
`!`
`interface TenGigabitEthernet1/0/1`
`no switchport`
`ip address 172.16.14.4 255.255.255.0`
`ip pim sparse-mode`
`ip ospf network point-to-point`
`ip ospf 1 area 0`
`!`
	`PE_device_3#` **`show running-config`**
	`hostname PE_device_3`
	`!`
	`vrf definition green`
	`rd 1:1`
	`!`
	`address-family ipv4`
	`route-target export 1:1`
	`route-target import 1:1`
	`exit-address-family`
	`!`
	`address-family ipv6`
	`route-target export 1:1`
	`route-target import 1:1`
	`exit-address-family`
	`!`
	`interface Loopback0`
	`ip address 172.16.255.103 255.255.255.255`
	`!`
	`interface Loopback1`
	`vrf forwarding green`
	`ip address 10.1.255.103 255.255.255.255`
	`!`
	`interface TenGigabitEthernet0/0/0`
	`ip address 172.16.111.103 255.255.255.0`
	`ip router isis`
	`cdp enable`
	`mpls ip`
	`isis network point-to-point`
	`!`
	`router isis`
	`net 49.0001.1720.1625.5103.00`
	`is-type level-2-only`
	`metric-style wide`
	`passive-interface Loopback0`
	`!`
	`router bgp 65002`
	`template peer-policy RR-PP`
	`route-reflector-client`
	`send-community both`
	`exit-peer-policy`
	`!`
	`template peer-session RR-PS`
	`remote-as 65002`
	`update-source Loopback0`
	`exit-peer-session`
	`!`
	`bgp log-neighbor-changes`
	`no bgp default ipv4-unicast`
	`neighbor 172.16.255.101 inherit peer-session RR-PS`
	`!`
	`address-family ipv4`
	`exit-address-family`
	`!`
	`!`
	`!`

VTEP 1	PE Device 3
!	
interface Vlan101	address-family vpnv4
vrf forwarding green	neighbor 172.16.255.101 activate
ip address 10.1.101.1 255.255.255.0	neighbor 172.16.255.101 send-community extended
!	neighbor 172.16.255.101 route-reflector-client
interface Vlan102	exit-address-family
vrf forwarding green	!
ip address 10.1.102.1 255.255.255.0	address-family vpnv6
!	neighbor 172.16.255.101 activate
interface Vlan901	neighbor 172.16.255.101 send-community extended
vrf forwarding green	neighbor 172.16.255.101 route-reflector-client
ip unnumbered Loopback1	exit-address-family
ipv6 enable	!
no autostate	address-family ipv4 vrf green
!	redistribute connected
interface nve1	exit-address-family
no ip address	!
source-interface Loopback1	address-family ipv6 vrf green
host-reachability protocol bgp	redistribute connected
member vni 10101 mcast-group 225.0.0.101	exit-address-family
member vni 50901 vrf green	!
member vni 10102 ingress-replication	end
!	!
router ospf 1	!
!	!
router bgp 65001	!
!	!
template peer-session RR-PS	!
remote-as 65001	!
update-source Loopback0	!
exit-peer-session	!
!	!
bgp log-neighbor-changes	!
no bgp default ipv4-unicast	!
neighbor 172.16.255.1 inherit peer-session RR-PS	!
!	!
address-family ipv4	!
exit-address-family	!
!	!
address-family l2vpn evpn	!
neighbor 172.16.255.1 activate	!
neighbor 172.16.255.1 send-community both	!
exit-address-family	!
!	!
address-family ipv4 vrf green	!
advertise l2vpn evpn	!
redistribute connected	!
redistribute static	!
exit-address-family	!
!	!
address-family ipv6 vrf green	!
advertise l2vpn evpn	!
redistribute connected	!
redistribute static	!
exit-address-family	!
!	!
ip pim rp-address 172.16.255.255	!
!	!
end	!
	!
VTEP_1#	PE_device_3#

The following examples provide sample outputs for **show** commands on the devices to verify external connectivity for the topology configured above:

VTEP 1

The following example shows the output for the **show bgp l2vpn evpn route-type** command for route type 5 on VTEP 1:

```
VTEP_1# show bgp l2vpn evpn route-type 5 0 10.1.255.103 32
BGP routing table entry for [5][1:1][0][32][10.1.255.103]/17, version 36
Paths: (1 available, best #1, table EVPN-BGP-Table)
  Not advertised to any peer
  Refresh Epoch 1
  65002
    172.16.254.6 (metric 3) (via default) from 172.16.255.1 (172.16.255.1)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Gateway Address: 0.0.0.0, VNI Label 50901, MPLS VPN
Label 0
      Extended Community: RT:1:1 ENCAP:8 Router MAC:0C75.BD67.EF48
      Originator: 172.16.255.6, Cluster list: 172.16.255.1
      rx pathid: 0, tx pathid: 0x0
      net: 0x7F84BB35A5C8, path: 0x7F84B913E010, pathext: 0x7F84BB54A8A8
      flags: net: 0x0, path: 0x3, pathext: 0x81
      Updated on May 21 2020 13:56:28 UTC
```

The following example shows the output for the **show bgp l2vpn evpn route-type** command for route type 2 on VTEP 1:

```
VTEP_1# show bgp l2vpn evpn route-type 2 0 44d3ca286cc1 10.1.101.2
BGP routing table entry for [2][172.16.254.4:101][0][48][44D3CA286CC1][32][10.1.101.2]/24,
 version 37
Paths: (1 available, best #1, table evi_101)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  Local
    :: (via default) from 0.0.0.0 (172.16.255.4)
      Origin incomplete, localpref 100, weight 32768, valid, sourced, local, best
      EVPN ESI: 00000000000000000000, Label1 10101, Label2 50901
      Extended Community: RT:1:1 RT:65001:101 ENCAP:8
        Router MAC:7C21.0DBD.9548
      Local irb vxlan vtep:
        vrf:green, l3-vni:50901
        local router mac:7C21.0DBD.9548
        core-irb interface:Vlan901
        vtep-ip:172.16.254.4
      rx pathid: 0, tx pathid: 0x0
      net: 0x7F84BB35A468, path: 0x7F84B913DF38, pathext: 0x7F84BB54A848
      flags: net: 0x0, path: 0x4000028000003, pathext: 0x81
      Updated on May 21 2020 14:00:49 UTC
```

The following example shows the output for the **show ip route vrf** command on VTEP 1:

```
VTEP_1# show ip route vrf green

Routing Table: green
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
```

```
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
C        10.1.101.0/24 is directly connected, Vlan101
L        10.1.101.1/32 is directly connected, Vlan101
C        10.1.102.0/24 is directly connected, Vlan102
L        10.1.102.1/32 is directly connected, Vlan102
B        10.1.255.101/32 [200/0] via 172.16.254.6, 00:06:25, Vlan901
B        10.1.255.103/32 [200/0] via 172.16.254.6, 00:05:54, Vlan901
```

Border VTEP

The following example shows the output for the **show bgp vpnv4 unicast all** command on border VTEP for the IP address of the external device:

```
Border_VTEP# show bgp vpnv4 uni all 10.1.255.103/32
BGP routing table entry for 1:1:10.1.255.103/32, version 9
Paths: (1 available, best #1, table green)
  Not advertised to any peer
  Refresh Epoch 1
  65002
    172.16.255.101 (via default) from 172.16.255.101 (172.16.255.101)
      Origin incomplete, localpref 100, valid, external, best
      Extended Community: RT:1:1
      Local vxlan vtep:
        vrf:green, vni:50901
        local router mac:0C75.BD67.EF48
        encap:8
        vtep-ip:172.16.254.6
        bdi:Vlan901
      mpls labels in/out nolabel/16
      rx pathid: 0, tx pathid: 0x0
      Updated on May 21 2020 13:48:09 UTC
```

The following example shows the output for the **show bgp l2vpn evpn route-type** command for route type 5 on border VTEP:

```
Border_VTEP# show bgp l2vpn evpn route-type 5 0 10.1.255.103 32
BGP routing table entry for [5][1:1][0][32][10.1.255.103]/17, version 32
Paths: (1 available, best #1, table EVPN-BGP-Table)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  65002, imported path from base
    172.16.255.101 (via default) from 172.16.255.101 (172.16.255.101)
      Origin incomplete, localpref 100, valid, external, best
      EVPN ESI: 00000000000000000000, Gateway Address: 0.0.0.0, local vtep: 172.16.254.6,
VNI Label 50901, MPLS VPN Label 16
      Extended Community: RT:1:1 ENCAP:8 Router MAC:0C75.BD67.EF48
      rx pathid: 0, tx pathid: 0x0
      net: 0x7FED704944D0, path: 0x7FED704A4CA0, pathext: 0x7FED6DA6E250, exp_net:
0x7FED6F812678
```

```
      flags: net: 0x0, path: 0x7, pathext: 0x81
      Updated on May 21 2020 13:48:09 UTC
```

The following example shows the output for the **show mpls forwarding-table** command on border VTEP:

```
Border_VTEP# show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         No Label   IPv4 VRF[V]      156           aggregate/green
17         Pop Label  172.16.106.101/32   \
                                       228           Te1/0/5    172.16.106.101
18         Pop Label  172.16.255.101/32   \
                                       0             Te1/0/5    172.16.106.101
```

The following example shows the output for the **show bgp vpnv4 unicast all** command on border VTEP for the IP address of host device 1:

```
Border_VTEP# show bgp vpnv4 uni all 10.1.101.2/32
BGP routing table entry for 1:1:10.1.101.2/32, version 10
Paths: (1 available, best #1, table green)
  Advertised to update-groups:
     1
  Refresh Epoch 4
  Local, imported path from [2][172.16.254.4:101][0][48][44D3CA286CC1][32][10.1.101.2]/24
(global)
    172.16.254.4 (metric 3) (via default) from 172.16.255.1 (172.16.255.1)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:1:1 ENCAP:8 Router MAC:7C21.0DBD.9548
      Originator: 172.16.255.4, Cluster list: 172.16.255.1
      Local vxlan vtep:
        vrf:green, vni:50901
        local router mac:0C75.BD67.EF48
        encap:8
        vtep-ip:172.16.254.6
        bdi:Vlan901
      Remote VxLAN:
        Topoid 0x9(vrf green)
        Remote Router MAC:7C21.0DBD.9548
        Encap 8
        Egress VNI 50901
        RTEP 172.16.254.4
      mpls labels in/out IPv4 VRF Aggr:16/nolabel
      rx pathid: 0, tx pathid: 0x0
      Updated on May 21 2020 13:52:30 UTC
```

Spine Switch

The following example shows the output for the **show bgp l2vpn evpn route-type** command for route type 5 on spine switch:

```
Spine_switch# show bgp l2vpn evpn route-type 5 0 10.1.255.103 32
BGP routing table entry for [5][1:1][0][32][10.1.255.103]/17, version 23
Paths: (1 available, best #1, table EVPN-BGP-Table)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  65002, (Received from a RR-client)
    172.16.254.6 (metric 2) (via default) from 172.16.255.6 (172.16.255.6)
```

```
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Gateway Address: 0.0.0.0, VNI Label 50901, MPLS VPN
Label 0
      Extended Community: RT:1:1 ENCAP:8 Router MAC:0C75.BD67.EF48
      rx pathid: 0, tx pathid: 0x0
      net: 0x7F54CC95FAB8, path: 0x7F54CCA542F8, pathext: 0x7F54CC9707B0
      flags: net: 0x0, path: 0x3, pathext: 0x81
      Updated on May 21 2020 13:54:20 UTC
```

The following example shows the output for the **show bgp l2vpn evpn route-type** command for route type 2 on spine switch:

```
Spine_switch# show bgp l2vpn evpn route-type 2 0 44d3ca286cc1 10.1.101.2
BGP routing table entry for [2][172.16.254.4:101][0][48][44D3CA286CC1][32][10.1.101.2]/24,
 version 24
Paths: (1 available, best #1, table EVPN-BGP-Table)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  Local, (Received from a RR-client)
    172.16.254.4 (metric 2) (via default) from 172.16.255.4 (172.16.255.4)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Label1 10101, Label2 50901
      Extended Community: RT:1:1 RT:65001:101 ENCAP:8
        Router MAC:7C21.0DBD.9548
      rx pathid: 0, tx pathid: 0x0
      net: 0x7F54CC95F958, path: 0x7F54CCA54220, pathext: 0x7F54CC970750
      flags: net: 0x0, path: 0x3, pathext: 0x81
      Updated on May 21 2020 13:58:41 UTC
```

PE Device 1

The following example shows the output for the **show bgp vpnv4 unicast all** command on PE device 1 for the IP address of host device 1:

```
PE_device_1# show bgp vpnv4 unicast all 10.1.255.103/32
BGP routing table entry for 1:1:10.1.101.2/32, version 14
Paths: (1 available, best #1, table green)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  65001
    172.16.255.6 (via default) from 172.16.255.6 (172.16.255.6)
      Origin incomplete, localpref 100, valid, external, best
      Extended Community: RT:1:1 ENCAP:8 Router MAC:7C21.0DBD.9548
      mpls labels in/out 22/16
      rx pathid: 0, tx pathid: 0x0
      Updated on May 21 2020 05:57:06 UTC
```

The following example shows the output for the **show ip route vrf** command on PE device 1:

```
PE_device_1# show ip route vrf green

Routing Table: green
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
```

```
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
B        10.1.101.0/24 [20/0] via 172.16.255.6, 00:28:09
B        10.1.101.1/32 [20/0] via 172.16.255.6, 00:28:09
B        10.1.101.2/32 [20/0] via 172.16.255.6, 00:23:17
B        10.1.102.0/24 [20/0] via 172.16.255.6, 00:28:09
B        10.1.102.1/32 [20/0] via 172.16.255.6, 00:28:09
C        10.1.255.101/32 is directly connected, Loopback1
B        10.1.255.103/32 [200/0] via 172.16.255.103, 00:28:09
```

PE Device 3

The following example shows the output for the **show bgp vpnv4 unicast all** command on PE device 3 for the IP address of host device 1:

```
PE_device_3# show bgp vpnv4 unicast all 10.1.101.2/32
BGP routing table entry for 1:1:10.1.101.2/32, version 14
Paths: (1 available, best #1, table green)
  Not advertised to any peer
  Refresh Epoch 1
  65001, (Received from a RR-client)
    172.16.255.101 (metric 10) (via default) from 172.16.255.101 (172.16.255.101)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:1:1 ENCAP:8 Router MAC:7C21.0DBD.9548
      mpls labels in/out nolabel/22
      rx pathid: 0, tx pathid: 0x0
      Updated on May 21 2020 05:56:46 UTC
```

The following example shows the output for the **show ip route vrf** command on PE device 3:

```
PE_device_3# show ip route vrf green

Routing Table: green
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
```

```
B        10.1.101.0/24 [200/0] via 172.16.255.101, 00:29:09
B        10.1.101.1/32 [200/0] via 172.16.255.101, 00:29:09
B        10.1.101.2/32 [200/0] via 172.16.255.101, 00:24:17
B        10.1.102.0/24 [200/0] via 172.16.255.101, 00:29:09
B        10.1.102.1/32 [200/0] via 172.16.255.101, 00:29:09
B        10.1.255.101/32 [200/0] via 172.16.255.101, 00:29:09
C        10.1.255.103/32 is directly connected, Loopback1
```

CHAPTER 7

Troubleshooting BGP EVPN VXLAN

Troubleshooting Scenarios for BGP EVPN VXLAN

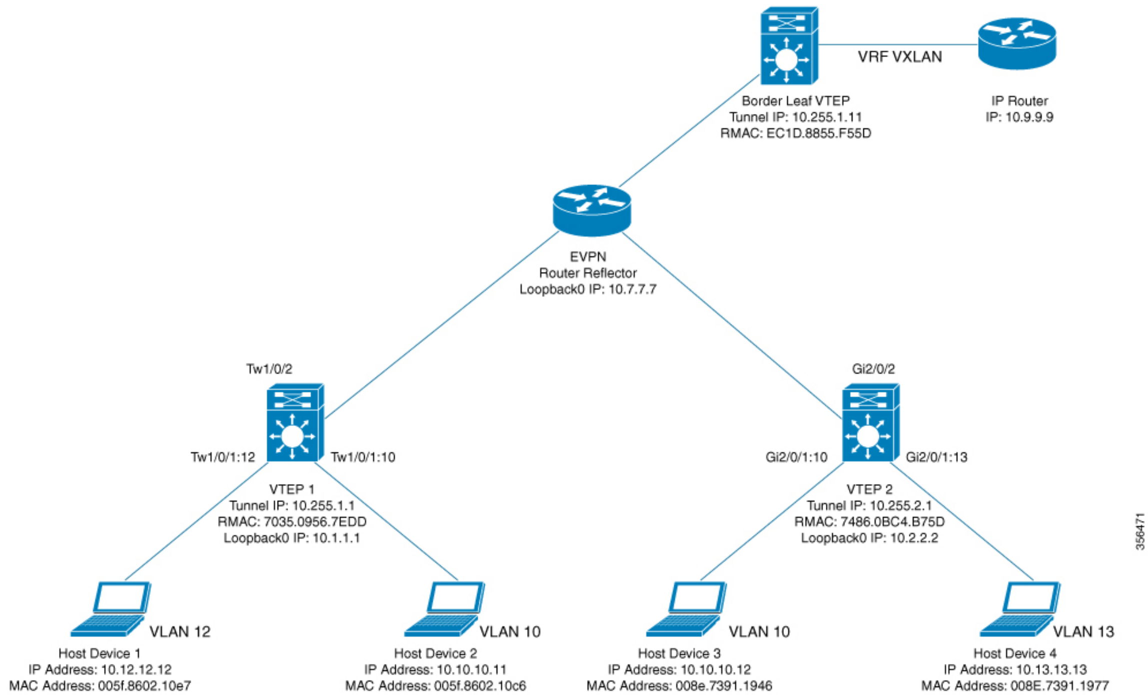
This document provides information about the various troubleshooting scenarios that are applicable to BGP EVPN VXLAN and how to troubleshoot each scenario.

In this troubleshooting document, comments have been added at the end of certain lines of the outputs of **show** commands. This has been done to highlight or explain a specific aspect of that line of output. If a comment begins in a new line, then it refers to the line of output that preceeds the comment. The following notation has been used throughout the document to highlight the comments inside the outputs of **show** commands:

```
<<— Text highlighted in this format inside a command's output represents a comment.
    This is done for explanation purpose only and is not part of the command's output.
```

The following is a sample EVPN VXLAN topology with two access facing VTEPs (VTEP 1 and VTEP 2) and a border leaf VTEP connected in a VXLAN network through an EVPN route reflector. Each of the access facing VTEPs has two host devices connected to it and the border leaf VTEP is connected to an external IP network. All the troubleshooting scenarios in this document are explained using this topology.

Figure 4: EVPN VXLAN Topology



The following are the various troubleshooting scenarios that apply to BGP EVPN VXLAN for the topology illustrated in the Figure 4: EVPN VXLAN Topology above:

- **Scenario 1:** Troubleshooting Broadcast, Unkown Unicast, Multicast traffic Forwarding
- **Scenario 2:** Troubleshooting Unicast Forwarding Between VTEPs in the Same VLAN Through a Layer 2 VNI
- **Scenario 3:** Troubleshooting Unicast Forwarding Between VTEPS in Different VLANs Through a Layer 3 VNI
- **Scenario 4:** Troubleshooting Unicast Forwarding Between a VXLAN Network and an IP Network

Troubleshooting Broadcast, Unkown Unicast, Multicast Traffic Forwarding

This scenario might occur when host device 2 attempts to learn the ARP for host device 3 in Figure 4: EVPN VXLAN Topology, on page 128. Perform the checks listed in the following table before troubleshooting BUM traffic forwarding:

Table 12: Scenario 1: Broadcast, Unkown Unicast, Multicast traffic Forwarding

Check to be Performed	Steps to Follow
Is the packet of broadcast type?	Check if the packet is a broadcast packet, such as an ARP broadcast packet.
Are the hosts in the same subnet or in different subnets?	Perform any of the following steps: • Check the host device. • Check the SVI configuration on the VTEP.
Has the remote MAC address been learned for unknown unicast traffic?	Run the **show platform software fed switch active matm macTable vlan** *vlan-id* command in privileged EXEC mode on the local VTEP and check if the MAC address of the remote host device is displaed in the output. If not, you have not yet learned the remote host device and it needs to be resolved.

BUM traffic is forwarded by a VTEP into the VXLAN Core using multicast routing. In order to follow the path of an ARP broadcast packet, you need to identify the multicast group that needs to be used to send this traffic into the core and to the other VTEPs. BUM traffic first arrives at the local Layer 2 interface. The traffic is encapsulated here and sent out using the multicast group that is sourced from the VXLAN Loopback interface.

Note Underlay multicast needs to be fully configured before troubleshooting BUM traffic forwarding for EVPN VXLAN.

To troubleshoot EVPN VXLAN BUM traffic forwarding, follow these steps:

1. Determine the MAC Address of the Local Host Device and the Multicast Group Used for ARP Tunneling, on page 129
2. Set Up Embedded Capture Towards the Core-Facing Interface, on page 130
3. Ping the Remote Host Device, on page 130
4. Verify that an ARP Request Has Been Received and a Multicast Route Has Been Built, on page 130
5. Confirm the Presence of ARP Request Replies in Embedded Capture, on page 131
6. Verify that the Encapsulated ARP Request is Leaving in a Multicast Group to a VXLAN UDP Destination Port, on page 131
7. Verify that the ARP Reply from Core Interface is Encapsulated in Unicast to a VXLAN UDP Destination Port, on page 132

Determine the MAC Address of the Local Host Device and the Multicast Group Used for ARP Tunneling

The following examples show how to verify the MAC address of the local host device and the multicast group that is used for tunneling the ARP broadcast request:

```
VTEP-1# show mac address-table address 005f.8602.10c6
Mac Address Table
-------------------------------------------

Vlan Mac Address Type     Ports
---- ----------- -------- -----
10 005f.8602.10c6 DYNAMIC Tw1/0/1    <<— MAC address of 10.10.10.11 is learnt here

VTEP-1# show run int nve 1
interface nve1
 no ip address
 source-interface Loopback999
 host-reachability protocol bgp
 member vni 10001 mcast-group 239.10.10.10    <<— Group is mapped to the VNI under NVE

VTEP-1# show run | s vlan conf
vlan configuration 10
 member evpn-instance 10 vni 10001    <<— VNI mapped under VLAN 10

VTEP-1# show l2vpn evpn evi
EVI   VLAN  Ether Tag  L2 VNI    Multicast     Pseudoport
----- ----- ---------- --------- ------------- ------------------
10    10    0          10001     239.10.10.10  Tw1/0/1:10
   <<— EVPN instance 10 is mapped to VLAN 10 and VNI 10001
       (Using multicast group 239.10.10.10 for Broadcast ecap tunnel)
<...snip...>
```

Set Up Embedded Capture Towards the Core-Facing Interface

The following example shows how to set up embedded capture towards the core-facing interface:

Note On a production network, use this command with a filter.

```
VTEP-1# show monitor capture 1 parameter
monitor capture 1 interface TwoGigabitEthernet1/0/2 BOTH
monitor capture 1 match any
monitor capture 1 buffer size 100
monitor capture 1 limit pps 1000
```

Ping the Remote Host Device

The following example shows how to ping the remote host device:

```
VTEP-1-HOST# ping 10.10.10.12     <<— sourced from Host machine 10.10.10.11
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.12, timeout is 2 seconds:
..!!!
```

Verify that an ARP Request Has Been Received and a Multicast Route Has Been Built

This step is to verify that there is multicast reachability between VTEPs using standard multicast validation. Underly multicast state is not permanent. If it is not in use, these S,G states will expire.

The following output confirms that an ARP request has been received and a multicast route has been built:

```
VTEP-1# show ip mroute 239.10.10.10 10.255.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
L - Local, P - Pruned, R - RP-bit set, F - Register flag,
T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
U - URD, I - Received Source Specific Host Report,
Z - Multicast Tunnel, z - MDT-data group sender,
Y - Joined MDT-data group, y - Sending to MDT-data group,
G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
Q - Received BGP S-A Route, q - Sent BGP S-A Route,
V - RD & Vector, v - Vector, p - PIM Joins on route,
x - VxLAN group, c - PFP-SA cache created entry
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(10.255.1.1, 239.10.10.10), 00:00:25/00:02:34, flags: FTx   <<— x flag set for VxLAN group
Incoming interface: Loopback999, RPF nbr 0.0.0.0       <<— Broadcast being encapsulated
                                                           into VXLAN tunnel IP
Outgoing interface list:
TwoGigabitEthernet1/0/2, Forward/Sparse, 00:00:23/00:03:06
   <<— Sending towards core to VTEP-2
(10.255.1.4, 239.10.10.10), 3d18h/00:02:25, flags: JTx <<— BUM traffic from VTEP-2 (if the
                                                            ARP request was from VTEP-2)
  Incoming interface: TwoGigabitEthernet1/0/2, RPF nbr 10.1.1.6
  Outgoing interface list:
    Tunnel0, Forward/Sparse-Dense, 3d18h/00:00:14    <<— Tunnel 0 is the VXLAN tunnel
                                                         used for decapsulation
```

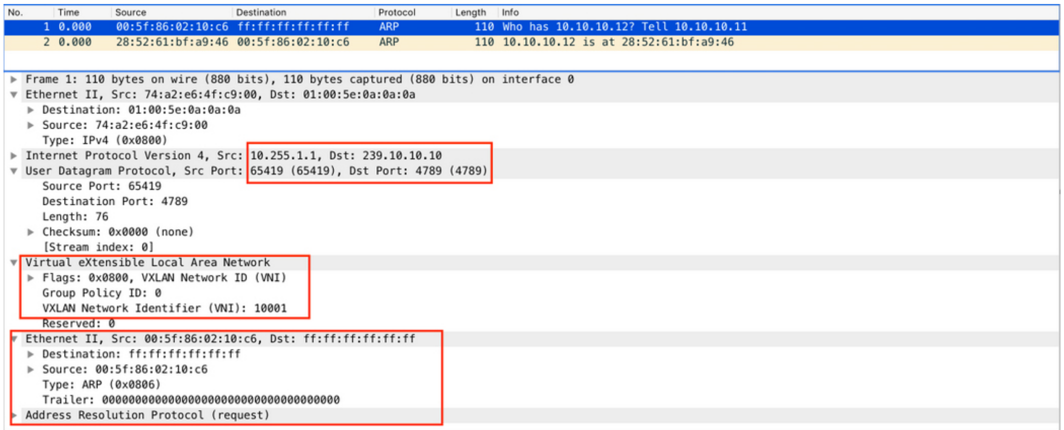
Confirm the Presence of ARP Request Replies in Embedded Capture

The following output confirms that the ARP request replies are present in embedded capture:

```
VTEP-1# show monitor capture 1 buffer display-filter "arp"
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

7 0.000018 00:5f:86:02:10:c6 -> ff:ff:ff:ff:ff:ff ARP 110 Who has 10.10.10.12? Tell
10.10.10.11
9 0.000022 28:52:61:bf:a9:46 -> 00:5f:86:02:10:c6 ARP 110 10.10.10.12 is at 28:52:61:bf:a9:46
```

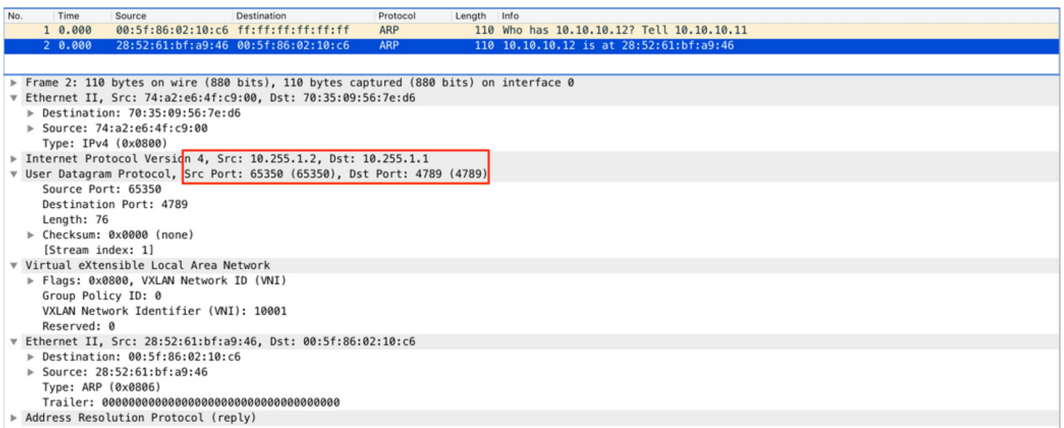
Verify that the Encapsulated ARP Request is Leaving in a Multicast Group to a VXLAN UDP Destination Port

The following image shows the ARP request leaving encapsulated in the multicast group 239.10.10.10, sourced from a VXLAN Loopback, to the VXLAN UDP destination port 4789 in the VNI 10001 and VLAN 10.



Verify that the ARP Reply from Core Interface is Encapsulated in Unicast to a VXLAN UDP Destination Port

The following image shows the ARP reply from core interface that is encapsulated in unicast, between VXLAN Loopbacks, to the VXLAN UDP destination port 4789 in the VNI 10001 and VLAN 10.



Once all of the above checks are verified, if there is still a problem with broadcast reachability, then repeat the checks on the remote VTEP.

Troubleshooting Unicast Forwarding Between VTEPs in the Same VLAN Through a Layer 2 VNI

This scenario might occur when host device 2 in VLAN 10 attempts to ping host device 3 that is also in VLAN 10. Perform the checks listed in the following table before troubleshooting unicast forwarding between VTEPs in the same VLAN through a Layer 2 VNI:

Table 13: Scenario 2: Troubleshooting Unicast Forwarding Between VTEPs in the Same VLAN Through a Layer 2 VNI

Check to be Performed	Steps to Follow
Has ARP been resolved on the local host for the Layer 2 adjacent remote host?	Run the **arp -a** command in privileged EXEC mode on the host device.
Do the hosts have the same subnet masks?	Perform any of the following steps: • Check the host device. • Check the SVI configuration on the VTEP.
Do you have the EVPN instance configured on your local VTEP?	Run the following commands in privileged EXEC mode on the VTEP: • **show run | section l2vpn** • **show run | section vlan config** • **show run interface nve** *interface-number*
Has the remote MAC address been learned in platform MATM in the same VLAN as the local host?	Run the **show platform software fed switch active matm macTable vlan** *vlan-id* command in privileged EXEC mode on the VTEP to check for the remote MAC addresses in the same VLAN.

To troubleshoot unicast forwarding between two VTEPs in the same VLAN using a Layer 2 VNI, follow these steps:

- Verify the provisioning of the EVPN VXLAN Layer 2 overlay network.
- Verify intra-subnet traffic movement in the EVPN VXLAN Layer 2 overlay network.

Verifying the Provisioning of an EVPN VXLAN Layer 2 Overlay Network

To verify the provisioning of an EVPN VXLAN Layer 2 overlay network, perform these checks:

1. Verify the Provisioning of the EVPN Instance in EVPN Manager, on page 133
2. Ensure that an NVE Peer is Present for the Layer 2 VNI, on page 135
3. Verify the Provisioning of the Layer 2 VNI in NVE Component, on page 135
4. Verify That the Layer 2 VNI VXLAN Tunnel Pseudoport is added to the Access VLAN in Layer 2 Forwarding Information Base (FIB), on page 136

Verify the Provisioning of the EVPN Instance in EVPN Manager

The following examples show how to verify that the EVPN instance is provisioned in the EVPN manager:

```
VTEP-1# show run | section l2vpn
l2vpn evpn instance 10 vlan-based
encapsulation vxlan
```

```
route-target export 10:1      <<– Import or export right route-targets

route-target import 10:2      <<– Import or export right route-targets

VTEP-1# show run | section vlan config
vlan configuration 10
member evpn-instance 10 vni 10001    <<– EVPN instance & VNI mapped to the VLAN

VTEP-1# show run interface nve1
interface nve1
source-interface Loopback999
host-reachability protocol bgp
member vni10001 mcast-group 239.10.10.10     <<– VNI added to NVE interface

VTEP-1# show run interface loopback 999
interface Loopback999
description VxLAN Loopback
ip address 10.255.1.1 255.255.255.255
```

Note Run the **show run** commands on VTEP 2 to verify its configuration, if required.

```
VTEP-1# show l2vpn evpn evi 10 detail <<– VLAN number and EVPN Instance number
                                          are not always the same, confirm which
                                          EVPN Instance maps to your VLAN
                                          with the show l2vpn evpn evi command
EVPN instance:    10 (VLAN Based)    <<– EVPN Instance number does map to the VLAN.
  RD:             10.1.1.1:10 (auto)
  Import-RTs:     10:2         <<– Importing VTEP-2 (if you are not seeing the prefix,
                                   check configuration for the right import/export statement
                                   under the l2vpn evpn instance)
  Export-RTs:     10:1
  Per-EVI Label:  none
  State:          Established
  Encapsulation:  vxlan
  Vlan:           10       <<– Layer 2 VLAN
    Ethernet-Tag: 0
    State:        Established <<– If State is not "Established", there
                                  could be a misconfiguration

    Core If:      Vlan99
    Access If:    Vlan10
    NVE If:       nve1
    RMAC:         7035.0956.7edd
    Core Vlan:    99
    L2 VNI:       10001           <<– Layer 2 VNI
    L3 VNI:       99999
    VTEP IP:      10.255.1.1
    MCAST IP:     239.10.10.10     <<– BUM Group for flooded traffic (Layer 2 learning, etc)

    VRF:          vxlan
    IPv4 IRB: Enabled
    IPv6 IRB: Enabled
    Pseudoports:
      TwoGigabitEthernet1/0/1 service instance 10
   <<– Layer 2 Access pseudoport (combination of Layer 2 port and service instance)
```

Note If only a Layer 2 overlay network has been configured for bridging, then the `Core If` , `Access If`, `RMAC`, `Core BD`, `L3 VNI`, and `VRF` fields do not show any values as they are not set.

```
VTEP-2# show l2vpn evpn evi 10 detail
EVPN instance:     10 (VLAN Based)
RD:                10.2.2.2:10 (auto)
Import-RTs:        10:1        <<— Importing VTEP-1 route-target
Export-RTs:        10:2
Per-EVI Label:     none
State:             Established
Encapsulation:     vxlan
Vlan:              10          <<— Layer 2 VLAN
  Ethernet-Tag:    0
  State:           Established
  Core If:         Vlan99
  Access If:       Vlan10
  NVE If:          nve1
  RMAC:            7486.0bc4.b75d
  Core Vlan:       99
  L2 VNI:          10001       <<— Layer 2 VNI
  L3 VNI:          99999
  VTEP IP:         10.255.2.1
  MCAST IP:        239.10.10.10
  VRF:             vxlan
  IPv4 IRB: Enabled
  IPv6 IRB: Enabled
  Pseudoports:
   GigabitEthernet2/0/1 service instance 10
   <<— Layer 2 Access pseudoport (combination of Layer 2 port and service instance)
```

Ensure that an NVE Peer is Present for the Layer 2 VNI

The following examples show how to check if an NVE peer is present for the Layer 2 VNI:

```
VTEP-1# show nve peers vni 10001     <<— This VNI is learned from "show l2vpn evpn evi"
Interface  VNI      Type Peer-IP          RMAC/Num_RTs   eVNI      state flags UP time
nve1       10001    L2CP 10.255.2.1       2              10001       UP   N/A 00:01:03
   <<— Layer 2 Control Plane (L2CP) peer for the VNI is an indicator that this is
       Layer 2 forwarding
   <<— Interface NVE1, L2CP, egress VNI are shown, state is UP for a time of 00:01:03

VTEP-2# show nve peers vni 10001
Interface  VNI      Type Peer-IP          RMAC/Num_RTs   eVNI      state flags UP time
nve1       10001    L2CP 10.255.1.1       3              10001       UP   N/A 00:47:2
   <<— Interface NVE1, L2CP, egress VNI are shown, state is UP for a time of 00:47:02
```

Verify the Provisioning of the Layer 2 VNI in NVE Component

The following example shows how to verify that the Layer 2 VNI is provisioned in the NVE component:

```
VTEP-1# show nve vni 10001 detail      <<— VNI 10001 is correlated to VLAN 10
                                          from show l2vpn evpn evi
Interface  VNI        Multicast-group VNI state  Mode  VLAN  cfg vrf
nve1       10001      239.10.10.10    Up         L2CP  10    CLI vxlan
```

```
   <<- state is UP, type is Layer 2 VNI (L2CP); VLAN 10 is mapped to VNI 10001

L2 VNI IPv6 IRB down reason:
BDI or associated L3 BDI's IPv6 addr un-configured
IPv6 topo_id disabled


L2CP VNI local VTEP info:      <<- Layer 2 VNI provisioning
VLAN: 10                       <<- Confirms that mapping is with VLAN 10
SVI if handler: 0x4D
Local VTEP IP: 10.255.1.1      <<- VxLAN Tunnel IP

Core IRB info:      <<- Layer 3 VPN provisioning (not required for troubleshooting
                        a scenario with pure Layer 2 VPN packet path
L3VNI: 99999
VRF name: vxlan
VLAN: 99
V4TopoID: 0x2
V6TopoID: 0xFFFF
Local VTEP IP: 10.255.1.1
SVI if handler: 0x50
SVI MAC: 7035.0956.7EDD

VNI Detailed statistics:
   Pkts In   Bytes In   Pkts Out  Bytes Out
         0          0 18158681548 27383291735556
```

Verify That the Layer 2 VNI VXLAN Tunnel Pseudoport is added to the Access VLAN in Layer 2 Forwarding Information Base (FIB)

The following examples show how to verify that the Layer 2 VXLAN tunnel pseudoport is added to the access VLAN in Layer 2 FIB:

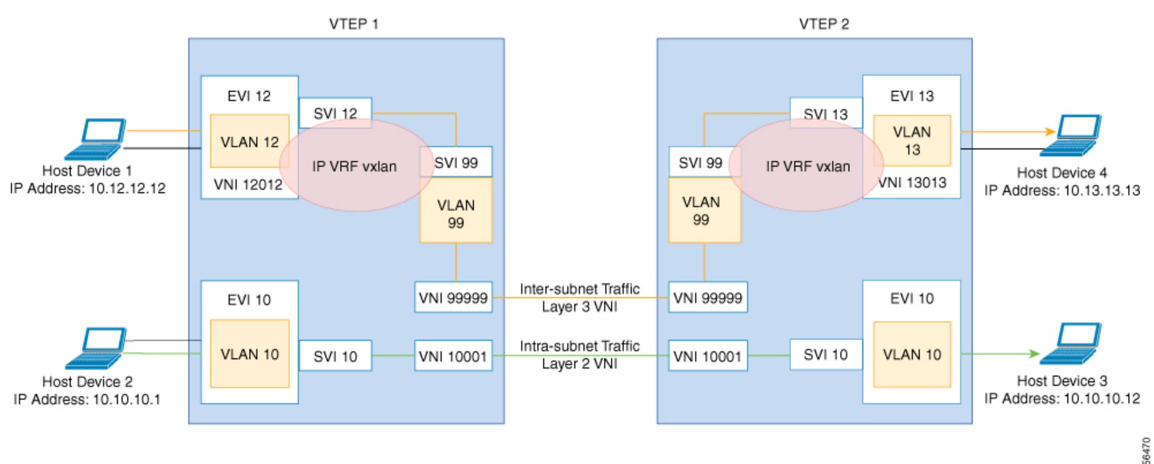
```
VTEP-1# show l2fib bridge-domain 10 detail    <<- Bridge-domain will be same as VLAN number
Bridge Domain : 10
  Reference Count : 14
  Replication ports count : 2
  Unicast Address table size : 3
  IP Multicast Prefix table size : 3

  Flood List Information :
    Olist: 5109, Ports: 2

  VxLAN Information :
    VXLAN_DEC nv1:10001:239.10.10.10

  Port Information :
    BD_PORT   Tw1/0/1:10      <<- Pseudoport has been added to bridge-domain:
                                  (physical port + the BD number for the VLAN)
    VXLAN_REP nv1:10001:239.10.10.10        <<- VXLAN Replication group

  Unicast Address table information :
    008e.7391.1946  VXLAN_CP  L:10001:10.255.1.1 R:10001:10.255.2.1

  IP Multicast Prefix table information :
    Source: *, Group: 224.0.0.0/24, IIF: Null, Adjacency: Olist: 5109, Ports: 2
    Source: *, Group: 224.0.1.39, IIF: Null, Adjacency: Olist: 5109, Ports: 2
    Source: *, Group: 224.0.1.40, IIF: Null, Adjacency: Olist: 5109, Ports: 2
```

```
VTEP-2# show l2fib bridge-domain 10 detail
Bridge Domain : 10
  Reference Count : 15
  Replication ports count : 2
  Unicast Address table size : 4
  IP Multicast Prefix table size : 3

  Flood List Information :
    Olist: 5109, Ports: 2

  VxLAN Information :
    VXLAN_DEC nv1:10001:239.10.10.10

  Port Information :
    BD_PORT   Gi2/0/1:10        <<- Pseudoport has been added to bridge-domain:
                                    (physical port + the BD number for the VLAN)
    VXLAN_REP nv1:10001:239.10.10.10          <<- VXLAN replication group

  Unicast Address table information :
    005f.8602.10c6  VXLAN_CP  L:10001:10.255.2.1 R:10001:10.255.1.1

  IP Multicast Prefix table information :
    Source: *, Group: 224.0.0.0/24, IIF: Null, Adjacency: Olist: 5109, Ports: 2
    Source: *, Group: 224.0.1.39, IIF: Null, Adjacency: Olist: 5109, Ports: 2
    Source: *, Group: 224.0.1.40, IIF: Null, Adjacency: Olist: 5109, Ports: 2
```

Verifying Intra-Subnet Traffic Movement in an EVPN VXLAN Layer 2 Overlay Network

The following figure illustrates the movement of traffic from host devices connected to VTEP 1 to host devices connected to VTEP 2:

Figure 5: Movement of traffic in an EVPN VXLAN network Through Layer 2 and Layer 3 VNIs



In the above figure, Layer 2 traffic moves from host device 2 to host device 3 through the Layer 2 VNI 10001. To verify the movement of intra-subnet traffic in the EVPN VXLAN Layer 2 overlay network, perform these checks:

1. Verify that the Local MAC Addresses Have Been Learned in IOS-MATM, on page 138
2. Verify that Both Local and Remote MAC Addresses are Learned in FED-MATM, on page 138

3. Confirm that the ICMP Echo Request Leaves VTEP 1 Encapsulated and Goes to a UDP Destination Port on VTEP 2, on page 139
4. Verify ARP for Local Host Devices, on page 139
5. Verify that the MAC Address Entries are Learned in SISF Device Tracking Table, on page 139
6. Verify that EVPN Manager Has Been Updated with the MAC Address Entries, on page 140
7. Verify that EVPN Manager Has Updated the MAC Routes into Layer 2 RIB, on page 141
8. Verify that Layer 2 RIB Has Updated BGP with the Local MAC Routes, and that BGP Has Updated Layer 2 RIB with the Remote MAC Routes, on page 141
9. Verify that the MAC Routes Learned from BGP and Updated to Layer 2 RIB are Also Updated to L2FIB, on page 143

Note Only MAC routes are considered while verifying the movement of intra-subnet traffic. MAC-IP routes are not applicable to bridged traffic.

Verify that the Local MAC Addresses Have Been Learned in IOS-MATM

The following examples show how to verify that the local MAC addresses have been learned in IOS-MATM:

```
VTEP-1# show mac address-table interface tw 1/0/1 vlan 10
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  10    005f.8602.10c6    DYNAMIC     Tw1/0/1     <<- IOS-MATM shows only
                                                      local MAC addresses

VTEP-2# show mac address-table interface g 2/0/1 vlan 10
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  10    008e.7391.1946    DYNAMIC     Gi2/0/1
```

Verify that Both Local and Remote MAC Addresses are Learned in FED-MATM

The following examples show how to verify that both local and remote MAC addresses are learned in FED-MATM:

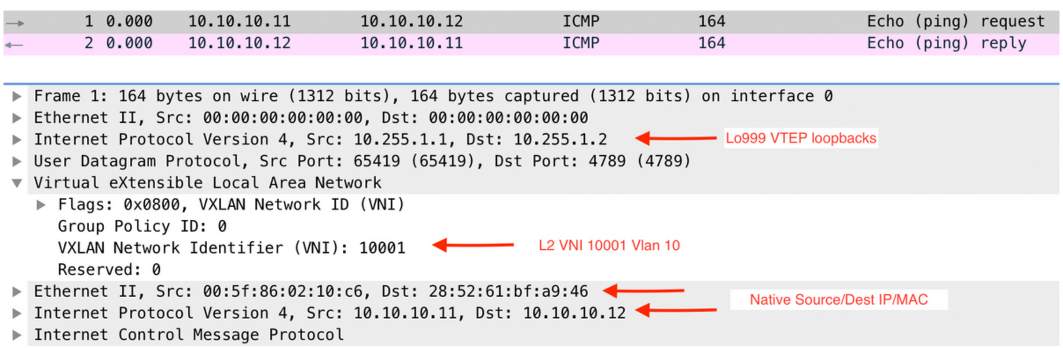
```
VTEP-1# show platform software fed switch active matm macTable vlan 10
VLAN   MAC                   Type   Seq#   EC_Bi  Flags  machandle
siHandle            riHandle            diHandle              *a_time   *e_time   ports
------------------------------------------------------------------------------------------

10     005f.8602.10c6         0x1      60      0       0  0x7efcc0d78fc8      0x7efcc0ca8b88
      0x0                    0x7efcc06cf9c8            300       144   TwoGigabitEthernet1/0/1

   <<- Local MAC address is displayed here
10     008e.7391.1946    0x1000001      0      0      64  0x7efcc0cafb38      0x7efcc0d7f628
```

```
      0x7ffa48c850b8      0x7efcc038cc18              0      144  RLOC 10.255.2.1 adj_id
135
   <<— Remote MAC address is displayed here

VTEP-2#sh platform software fed switch active matm macTable vlan 10
VLAN   MAC                    Type  Seq#   EC_Bi  Flags  machandle           siHandle
      riHandle            diHandle               *a_time  *e_time  ports
------------------------------------------------------------------------------------------
10     005f.8602.10c6   0x1000001     0      0      64  0x7fcec4e977d8      0x7fcec4e93ae8
      0x7fcec4e93308      0x7fcec430a3d8              0         0  RLOC 10.255.1.1 adj_id
64
   <<— Remote MAC address is displayed here
10     008e.7391.1946         0x1     46      0      0  0x7fcec4c6a248      0x7fcec4c20698
      0x0                 0x7fcec4611438             300       126  GigabitEthernet2/0/1

   <<— Local MAC address is displayed here
```

Confirm that the ICMP Echo Request Leaves VTEP 1 Encapsulated and Goes to a UDP Destination Port on VTEP 2

The following image confirms that the ICMP echo request leaves VTEP 1 encapsulated and goes to a UDP destination port on VTEP 2 through the loopback interface Lo999 and the Layer 2 VNI 10001:

Figure 6:



Verify ARP for Local Host Devices

The following examples show how to verify ARP for local host devices:

```
VTEP-1# show ip arp vrf vxlan 10.10.10.11
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.10.10.11              2  005f.8602.10c6  ARPA   Vlan10

VTEP-2# show ip arp vrf vxlan 10.10.10.12
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.10.10.12              4  008e.7391.1946  ARPA   Vlan10
```

Verify that the MAC Address Entries are Learned in SISF Device Tracking Table

The following examples show how to verify that the MAC addresses are learned in SISF device tracking table:

```
VTEP-1# show device-tracking database mac   <<— Only Local MAC addresses are seen
                                                in SISF device tracking table
 MAC               Interface      vlan prlvl      state           time left policy
 005f.8602.10c6    Tw1/0/1          10 NO TRUST   MAC-REACHABLE  347 s     evpn-sisf-policy
   <<— MAC, REACH, and EVPN type SISF policy are displayed

VTEP-2# show device-tracking database mac   <<— Only Local MAC addresses are seen
                                                in SISF device tracking table
 MAC               Interface      vlan prlvl      state           time left policy
 008e.7391.1946    Gi2/0/1          10 NO TRUST   MAC-REACHABLE  164 s     evpn-sisf-policy
   <<— MAC, REACH, and EVPN type SISF policy are displayed
```

Verify that EVPN Manager Has Been Updated with the MAC Address Entries

EVPN manager learns local MAC addresses and adds them to Layer 2 RIB. EVPN Manager also learns the remote MAC addresses from Layer 2 RIB, but the entries are only used for processing MAC mobility.

The following examples show how to verify that EVPN manager has been updated with the MAC addresses:

```
VTEP-1# show l2vpn evpn mac evi 10
MAC Address    EVI   VLAN  ESI                      Ether Tag  Next Hop
-------------- ----- ----- ------------------------ ---------- ---------------
005f.8602.10c6 10    10    0000.0000.0000.0000.0000 0          Tw1/0/1:10
   <<— MAC Addresss learned by EVPN Manager. States look correct
008e.7391.1946 10    10    0000.0000.0000.0000.0000 0          10.255.2.1

VTEP-1#sh l2vpn evpn mac evi 10 detail
MAC Address:               005f.8602.10c6      <<— Local MAC address
EVPN Instance:             10       <<— EVPN Instance
Vlan:                      10       <<— VLAN
Ethernet Segment:          0000.0000.0000.0000.0000
Ethernet Tag ID:           0
Next Hop(s):               TwoGigabitEthernet1/0/1 service instance 10<<— Local interface
                                                                        or local instance
VNI:                       10001       <<— VNI Label
Sequence Number:           0
MAC only present:          Yes
MAC Duplication Detection: Timer not running

MAC Address:               008e.7391.1946      <<— Remote MAC Address
EVPN Instance:             10        <<— EVPN Instance
Vlan:                      10        <<— VLAN
Ethernet Segment:          0000.0000.0000.0000.0000
Ethernet Tag ID:           0
Next Hop(s):               10.255.2.1       <<— Remote VTEP-2 Tunnel Loopback
Local Address:             10.255.1.1       <<— Local VTEP-1 Tunnel Loopback
VNI:                       10001       <<— VNI Label
Sequence Number:           0
MAC only present:          Yes
MAC Duplication Detection: Timer not running

VTEP-2# show l2vpn evpn mac evi 10
MAC Address    EVI   VLAN  ESI                      Ether Tag  Next Hop
-------------- ----- ----- ------------------------ ---------- ---------------
005f.8602.10c6 10    10    0000.0000.0000.0000.0000 0          10.255.1.1
008e.7391.1946 10    10    0000.0000.0000.0000.0000 0          Gi2/0/1:10
```

```
VTEP-2#sh l2vpn evpn mac evi 10 detail
MAC Address:               005f.8602.10c6     <<— Remote MAC address
EVPN Instance:             10       <<— EVPN Instance
Vlan:                      10       <<— VLAN
Ethernet Segment:          0000.0000.0000.0000.0000
Ethernet Tag ID:           0
Next Hop(s):               10.255.1.1     <<— Remote VTEP-1 Tunnel Loopback
Local Address:             10.255.2.1     <<— Local VTEP-2 Tunnel Loopback
VNI:                       10001          <<— VNI Label
Sequence Number:           0
MAC only present:          Yes
MAC Duplication Detection: Timer not running

MAC Address:               008e.7391.1946       <<— Remote MAC adress
EVPN Instance:             10        <<— EVPN Instance
Vlan:                      10        <<— VLAN
Ethernet Segment:          0000.0000.0000.0000.0000
Ethernet Tag ID:           0
Next Hop(s):               GigabitEthernet2/0/1 service instance 10  <<— Local interface
                                                                         or local instance
VNI:                       10001       <<— VNI Label
Sequence Number:           0
MAC only present:          Yes
MAC Duplication Detection: Timer not running
```

Verify that EVPN Manager Has Updated the MAC Routes into Layer 2 RIB

Layer 2 RIB learns local MAC addresses from EVPN manager and updates BGP and Layer 2 FIB with them. Layer 2 RIB also learns remote MAC addresses from BGP and updates EVPN manager and Layer 2 FIB with them. Layer 2 RIB needs both local and remote MAC addresses in order to update BGP and Layer 2 FIB.

The following examples show how to verify that EVPN manager has updated the MAC routes into Layer 2 RIB:

```
VTEP-1# show l2route evpn mac
  EVI       ETag  Prod     Mac Address                  Next Hop(s) Seq Number
----- ---------- ----- -------------- ---------------------------- ----------
   10          0 L2VPN 005f.8602.10c6                   Tw1/0/1:10          0
   <<— Local prefix was added by EVPN Manager (Layer 2 VPN) into Layer 2 RIB
   10          0   BGP 008e.7391.1946           V:10001 10.255.2.1          0
   <<— Remote prefix was added by BGP into Layer 2 RIB

VTEP-2# show l2route evpn mac
  EVI       ETag  Prod     Mac Address                  Next Hop(s) Seq Number
----- ---------- ----- -------------- ---------------------------- ----------
   10          0   BGP 005f.8602.10c6           V:10001 10.255.1.1          0
   <<— Remote prefix was added by BGP into Layer 2 RIB
   10          0 L2VPN 008e.7391.1946                   Gi2/0/1:10          0
   <<— Local prefix was added by EVPN Manager (Layer 2 VPN) into Layer 2 RIB
```

Verify that Layer 2 RIB Has Updated BGP with the Local MAC Routes, and that BGP Has Updated Layer 2 RIB with the Remote MAC Routes

The following examples show how top verify that Layer 2 RIB has updated BGP with the local MAC routes and that BGP has updated Layer 2 RIB with the remote MAC routes:

```
VTEP-1# show bgp l2vpn evpn route-type 2 0 005f860210c6 *
```

```
   <<— Route-type is 2, Ethernet tag = 0, Local MAC address is in
       undelimited format, and * specifies to omit IP address
BGP routing table entry for [2][10.1.1.1:10][0][48][005F860210C6][0][*]/20, version 249
Paths: (1 available, best #1, table evi_10) <<— Added to BGP from EVPN Manager
                                                provisioning in l2vpn evi context
  Advertised to update-groups:
     2
  Refresh Epoch 1
  Local
    :: (via default) from 0.0.0.0 (10.1.1.1)     <<— Locally Advertised by VTEP-1,
                                                     (:: indicates local)
      Origin incomplete, localpref 100, weight 32768, valid, sourced, local, best
      EVPN ESI: 00000000000000000000, Label1 10001   <<— VNI ID is 10001 for VLAN 10
      Extended Community: RT:10:1 ENCAP:8    <<— RT 10:1 (local RT), Encap type 8 is VXLAN
      Local irb vxlan vtep:
        vrf:vxlan, l3-vni:99999
        local router mac:7035.0956.7EDD
        core-irb interface:Vlan99
        vtep-ip:10.255.1.1
      rx pathid: 0, tx pathid: 0x0

VTEP-1# show bgp l2vpn evpn route-type 2 0 008e73911946 *
   <<— Route-type is 2, Ethernet tag = 0, Remote MAC address is in
       undelimited format, and * specifies to omit IP address
BGP routing table entry for [2][10.1.1.1:10][0][48][008e73911946][0][*]/20, version 253
Paths: (1 available, best #1, table evi_10)    <<— EVPN instance BGP table for VLAN 10
  Not advertised to any peer
  Refresh Epoch 1
  Local, imported path from [2][10.2.2.2:10][0][48][008e73911946][0][*]/20 (global)
   <<— From VTEP-2, RD is 10.2.2.2:10, MAC length is 48, [*] indicates MAC only
    10.255.2.1 (metric 2) (via default) from 10.2.2.2 (10.2.2.2)
   <<— Next hop of VTEP-2 Lo999, learned from RR 10.2.2.2
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Label1 10001    <<— VNI ID 10001 for VLAN 10
      Extended Community: RT:10:2 ENCAP:8     <<— Layer 2 VPN Route-Target 10:2
                                                  Encap type 8 is VXLAN
      Originator: 10.2.2.2, Cluster list: 10.2.2.2
      rx pathid: 0, tx pathid: 0x0

BGP routing table entry for [2][10.2.2.2:10][0][48][008e73911946][0][*]/20, version 251
Paths: (1 available, best #1, table EVPN-BGP-Table)
  Not advertised to any peer
  Refresh Epoch 1
  Local
    10.255.2.1 (metric 2) (via default) from 10.2.2.2 (10.2.2.2)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Label1 10001
      Extended Community: RT:10:2 ENCAP:8
      Originator: 10.2.2.2, Cluster list: 10.2.2.2
      rx pathid: 0, tx pathid: 0x0

VTEP-2# show bgp l2vpn evpn route-type 2 0 008e73911946 *
   <<— Route-type is 2, Ethernet tag = 0, Local MAC address is in
       undelimited format, and * specifies to omit IP address
BGP routing table entry for [2][10.2.2.2:10][0][48][008e73911946][0][*]/20, version 292
Paths: (1 available, best #1, table evi_10)
  Advertised to update-groups:
     2
  Refresh Epoch 1
  Local
    :: (via default) from 0.0.0.0 (10.2.2.2)     <<— Locally Advertised by VTEP-2,
                                                     (:: indicates local)
```

```
      Origin incomplete, localpref 100, weight 32768, valid, sourced, local, best
      EVPN ESI: 00000000000000000000, Label1 10001    <<— VNI ID 10001 for VLAN 10
      Extended Community: RT:10:2 ENCAP:8    <<— RT 10:2 (local RT), Encap type 8 is VXLAN
      Local irb vxlan vtep:
        vrf:vxlan, l3-vni:99999
        local router mac:7486.0BC4.B75D
        core-irb interface:Vlan99
        vtep-ip:10.255.2.1
      rx pathid: 0, tx pathid: 0x0

VTEP-2# show bgp l2vpn evpn route-type 2 0 005f860210c6 *
   <<— Route-type is 2, Ethernet tag = 0, Remote MAC address is in
       undelimited format, and * specifies to omit IP address
BGP routing table entry for [2][10.1.1.1:10][0][48][005F860210C6][0][*]/20, version 312
Paths: (1 available, best #1, table EVPN-BGP-Table)
  Not advertised to any peer
  Refresh Epoch 7
  Local
    10.255.1.1 (metric 2) (via default) from 10.2.2.2 (10.2.2.2)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Label1 10001
      Extended Community: RT:10:1 ENCAP:8
      Originator: 10.1.1.1, Cluster list: 10.2.2.2
      rx pathid: 0, tx pathid: 0x0

BGP routing table entry for [2][10.2.2.2:10][0][48][005F860210C6][0][*]/20, version 314
Paths: (1 available, best #1, table evi_10)    <<— EVPN instance BGP table for VLAN 10
  Not advertised to any peer
  Refresh Epoch 7
  Local, imported path from [2][10.1.1.1:10][0][48][005F860210C6][0][*]/20 (global)
   <<— From VTEP-2, RD is 10.2.2.2:10, MAC length is 48, [*] indicates MAC only
   <<— From VTEP-1, RD is 10.1.1.1:10, MAC length is 48, [*] indicates MAC only
    10.255.1.1 (metric 2) (via default) from 10.2.2.2 (10.2.2.2)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Label1 10001     <<— VNI ID 10001 for VLAN 10
      Extended Community: RT:10:1 ENCAP:8     <<— Layer 2 VPN Route-Target 10:1
                                                  Encap type 8 is VXLAN
      Originator: 10.1.1.1, Cluster list: 10.2.2.2
      rx pathid: 0, tx pathid: 0x0
```

Verify that the MAC Routes Learned from BGP and Updated to Layer 2 RIB are Also Updated to L2FIB

The following examples show how to verify that the MAC routes that are learned from BGP and updated to Layer 2 RIB are also updated to Layer 2 FIB:

```
VTEP-2# show l2fib bridge-domain 10 detail
Bridge Domain : 10
  Reference Count : 15
  Replication ports count : 2
  Unicast Address table size : 4
  IP Multicast Prefix table size : 3

  Flood List Information :
    Olist: 5109, Ports: 2

  VxLAN Information :
    VXLAN_DEC nv1:10001:239.10.10.10

  Port Information :
    BD_PORT   Gi2/0/1:10
```

```
    VXLAN_REP nv1:10001:239.10.10.10

  Unicast Address table information :
    005f.8602.10c6  VXLAN_CP  L:10001:10.255.2.1 R:10001:10.255.1.1
   <<— Remote MAC address is learned (local MAC address is not expected to be present)

  IP Multicast Prefix table information :
    Source: *, Group: 224.0.0.0/24, IIF: Null, Adjacency: Olist: 5109, Ports: 2
    Source: *, Group: 224.0.1.39, IIF: Null, Adjacency: Olist: 5109, Ports: 2
    Source: *, Group: 224.0.1.40, IIF: Null, Adjacency: Olist: 5109, Ports: 2

VTEP-1# show l2fib bridge-domain 10 detail
Bridge Domain : 10
  Reference Count : 14
  Replication ports count : 2
  Unicast Address table size : 3
  IP Multicast Prefix table size : 3

  Flood List Information :
    Olist: 5109, Ports: 2

  VxLAN Information :
    VXLAN_DEC nv1:10001:239.10.10.10

  Port Information :
    BD_PORT   Tw1/0/1:10
    VXLAN_REP nv1:10001:239.10.10.10

  Unicast Address table information :
    008e.7391.1946  VXLAN_CP  L:10001:10.255.1.1 R:10001:10.255.2.1
   <<— Remote MAC address is learned (local MAC address is not expected to be present)

  IP Multicast Prefix table information :
    Source: *, Group: 224.0.0.0/24, IIF: Null, Adjacency: Olist: 5109, Ports: 2
    Source: *, Group: 224.0.1.39, IIF: Null, Adjacency: Olist: 5109, Ports: 2
    Source: *, Group: 224.0.1.40, IIF: Null, Adjacency: Olist: 5109, Ports: 2
```

Note Only remote MAC routes are displayed in the output.

Troubleshooting Unicast Forwarding Between VTEPS in Different VLANs Through a Layer 3 VNI

This scenario might occur when host device 1 in VLAN 12 attempts to ping host device 4 in VLAN 13. Perform the checks listed in the following table before troubleshooting unicast forwarding between VTEPs in different VLANs through a Layer 3 VNI:

Table 14: Scenario 3: Troubleshooting Unicast Forwarding Between VTEPS in Different VLANs Through a Layer 3 VNI

Check to be Performed	Steps to Follow
Are the source and destination host devices in different subnets?	Check the subnet of the local host device and compare it against the subnet of the remote host device.

Check to be Performed	Steps to Follow
Do you have an SVI interface configured for the remote subnet?	Run the **show ip interface brief | excluse unassigned** command in privileged EXEC mode on the VTEP.
Do you have the EVPN instance configured on your local VTEP?	Run the following commands in privileged EXEC mode on the VTEP: • **show run | section l2vpn** • **show run | section vlan config** • **show run interface nve** *interface-number*

To troubleshoot unicast forwarding between two VTEPs in different VLANs using a Layer 3 VNI, follow these steps:

- Verify the provisioning of the EVPN VXLAN Layer 3 overlay network.
- Verify inter-subnet traffic movement and symmetric IRB in the EVPN VXLAN Layer 3 overlay network.

Verifying the Provisioning of an EVPN VXLAN Layer 3 Overlay Network

To verify the provisioning of an EVPN VXLAN Layer 3 overlay network, perform these checks:

1. Verify that the Access SVIs, Core SVIs, and NVE Interfaces are Up, on page 145
2. Verify that the IP VRF is Provisioned with the Correct SVIs, Stitching Route-Targets, and Route Distinguisher, on page 146
3. Verify that Both Layer 2 and Layer 3 VNIs are provisioned in the VRF and are UP, on page 147
4. Verify that EVPN Manager is Updated from the NVE with all the Layer 2 and IRB Attributes, on page 148
5. Verify that the Remote Layer 3 VNI Details are Learned on Each VTEP, on page 149
6. Verify that the Layer 3 VNI Tunnel Pseudoport is Installed into Layer 2 FIB in the Core VLAN, on page 149

Verify that the Access SVIs, Core SVIs, and NVE Interfaces are Up

The following examples show how to verify that the access SVIs, core SVIs, and NVE interfaces are up:

```
VTEP-1# show ip interface brief
Interface               IP-Address      OK? Method Status          Protocol
Vlan10                  10.10.10.1      YES NVRAM  up              up
Vlan12                  10.12.12.1      YES NVRAM  up              up  <<— Access Interface
Vlan99                  10.255.1.1      YES unset  up              up  <<— Core Interface
   <<— If protocol status for the core interface is down, run the no autostate command
Loopback0               10.1.1.1        YES NVRAM  up              up
Loopback999             10.255.1.1      YES NVRAM  up              up
Tunnel0                 10.255.1.1      YES unset  up              up
```

```
Tunnel1                 10.1.1.5        YES unset  up              up
nve1                    unassigned      YES unset  up              up

VTEP-2# show ip interface brief
Interface               IP-Address      OK? Method Status          Protocol
Vlan10                  10.10.10.1      YES NVRAM  up              up
Vlan13                  10.13.13.1      YES NVRAM  up              up  <<— Access Interface
Vlan99                  10.255.2.1      YES unset  up              up  <<— Core Interface
   <<— If protocol status for the core interface is down, run the no autostate command
Loopback0               10.2.2.2        YES NVRAM  up              up
Loopback999             10.255.2.1      YES NVRAM  up              up
Tunnel0                 10.255.2.1      YES unset  up              up
Tunnel1                 10.1.1.10       YES unset  up              up
```

Verify that the IP VRF is Provisioned with the Correct SVIs, Stitching Route-Targets, and Route Distinguisher

The following examples show how to verify that the IP VRF is provisioned with the correct SVIs, stitching route-targets, and route distinguisher:

```
VTEP-1# show run vrf vxlan    <<— vxlan is the name of the VRF
vrf definition vxlan
rd 10.255.1.1:1
!
address-family ipv4
 route-target export 10.255.1.1:1 stitching     <<— Exporting local route-target
 route-target import 10.255.2.1:1 stitching     <<— Importing VTEP-2 route-target

VTEP-1# show ip vrf vxlan    <<— vxlan is the name of the VRF
  Name                              Default RD            Interfaces
  vxlan                             10.255.1.1:1          Vl10
                                                          Vl12
                                                          Vl99

VTEP-1# show ip vrf detail vxlan    <<— vxlan is the name of the VRF
VRF vxlan (VRF Id = 2); default RD 10.255.1.1:1; default VPNID <not set>
New CLI format, supports multiple address-families
Flags: 0x180C
Interfaces:
Vl10 Vl12 Vl99
Address family ipv4 unicast (Table ID = 0x2):    <<— Table 2 maps to VRF vxlan,
                                                     also found in BPG VPNv4 table
Flags: 0x0
No Export VPN route-target communities
No Import VPN route-target communities
Export VPN route-target stitching communities
   <<— VRF is using stitching route-targets. VTEPs must
       import each other's targets (same as Layer 3 VPN)
RT:10.255.1.1:1
Import VPN route-target stitching communities
RT:10.255.2.1:1
No import route-map
No global export route-map
No export route-map
VRF label distribution protocol: not configured
VRF label allocation mode: per-prefix

VTEP-2# show ip vrf vxlan    <<— vxlan is the name of the VRF
  Name                              Default RD            Interfaces
```

```
  vxlan                             10.255.2.1:1          Vl10
                                                          Vl13
                                                          Vl99

VTEP-2# show ip vrf detail vxlan    <<— vxlan is the name of the VRF
VRF vxlan (VRF Id = 2); default RD 10.255.2.1:1; default VPNID <not set>
New CLI format, supports multiple address-families
Flags: 0x180C
Interfaces:
Vl10 Vl13 Vl99
Address family ipv4 unicast (Table ID = 0x2):  <<— Table 2 maps to VRF vxlan,
                                                   also found in BPG VPNv4 table
Flags: 0x0
No Export VPN route-target communities
No Import VPN route-target communities
Export VPN route-target stitching communities
   <<— VRF is using stitching route-targets. VTEPs must
       import each other's targets (same as Layer 3 VPN)
RT:10.255.2.1:1
Import VPN route-target stitching communities
RT:10.255.1.1:1
No import route-map
No global export route-map
No export route-map
VRF label distribution protocol: not configured
VRF label allocation mode: per-prefix
```

Verify that Both Layer 2 and Layer 3 VNIs are provisioned in the VRF and are UP

The following examples show how to verify that both Layer 2 and Layer 3 VNIs are provisioned in the VRF and are up:

```
VTEP-1# show run | section vlan config
vlan configuration 99     <<— VNI is a member of VRF vxlan, not of EVPN instance
 member vni99999

VTEP-1# show run interface vlan 99
interface Vlan99
 description connected to L3_VNI_99999
 vrf forwarding vxlan
 ip unnumbered Loopback999

VTEP-1# show run interface nve 1
no ip address
 source-interface Loopback999
 host-reachability protocol bgp
 member vni 99999 vrf vxlan      <<— VNI tied to the VRF under NVE interface
 member vni 12012 mcast-group 239.12.12.12 <<— VNI tied to the NVE

VTEP-1# show run | section l2vpn
l2vpn evpn instance 12 vlan-based
 encapsulation vxlan
 route-target export 12:1    <<— Remote VTEP is NOT importing this route target,
                                 as it does not have the VLAN or VNI on its end
 route-target import 12:1
 no auto-route-target

VTEP-1# show run | section vlan config
```

```
vlan configuration 12
 member evpn-instance 12 vni 12012  <<— EVPN instance or VNI associated to the VLAN

VTEP-1# show nve vni
Interface  VNI        Multicast-group VNI state  Mode  VLAN  cfg vrf
nve1       10001      239.10.10.10    Up         L2CP  10    CLI vxlan
nve1       12012      239.12.12.12    Up         L2CP  12    CLI vxlan <<— Layer 2 VNI
nve1       99999      N/A             Up         L3CP  99    CLI vxlan <<— Layer 3 VNI

VTEP-2# show nve vni
Interface  VNI        Multicast-group VNI state  Mode  VLAN  cfg vrf
nve1       13013      239.13.13.13    Up         L2CP  13    CLI vxlan <<— Layer 2 VNI
nve1       10001      239.10.10.10    Up         L2CP  10    CLI vxlan
nve1       99999      N/A             Up         L3CP  99    CLI vxlan <<— Layer 3 VNI
```

Verify that EVPN Manager is Updated from the NVE with all the Layer 2 and IRB Attributes

The following examples show how to verify that EVPN manager is updated from the NVE with all the Layer 2 and IRB attributes:

```
VTEP-1# show l2vpn evpn evi
EVI   VLAN  Ether Tag  L2 VNI    Multicast     Pseudoport
----- ----- ---------- --------- ------------- ------------------
12    12    0          12012     239.12.12.12  Tw1/0/1:12
   <<— See which EVPN instance maps to the VLAN. The VLAN
       or EVPN instance values are not always the same
<...snip...>

VTEP-1# show l2vpn evpn evi 12 detail
EVPN instance:    12 (VLAN Based)
  RD:             10.1.1.1:12 (auto)
  Import-RTs:     12:1
  Export-RTs:     12:1
  Per-EVI Label:  none
  State:          Established
  Encapsulation:  vxlan
  Vlan:           12    <<— VLAN Layer 2 VNI
    Ethernet-Tag: 0
    State:        Established
    Core If:      Vlan99    <<— Interface handling IP VRF forwarding
    Access If:    Vlan12
    NVE If:       nve1
    RMAC:         7035.0956.7edd  <<— RMAC is the BIA of SVI 99 Core interface
    Core Vlan:    99
    L2 VNI:       12012
    L3 VNI:       99999
    VTEP IP:      10.255.1.1   <<— Local Tunnel endpoint IP address
    MCAST IP:     239.12.12.12
    VRF:          vxlan    <<— IP VRF for Layer 3 VPN
    Pseudoports:
      TwoGigabitEthernet1/0/1 service instance 12

VTEP-2# show l2vpn evpn evi
EVI   VLAN  Ether Tag  L2 VNI    Multicast     Pseudoport
----- ----- ---------- --------- ------------- ------------------
13    13    0          13013     239.13.13.13  Gi2/0/1:13
   <<— See which EVPN instance maps to the VLAN. The VLAN
       or EVPN instance values are not always the same
```

```
VTEP-2# show l2vpn evpn evi 13 detail
EVPN instance:    13 (VLAN Based)
  RD:             10.2.2.2:13 (auto)
  Import-RTs:     13:2
  Export-RTs:     13:2
  Per-EVI Label:  none
  State:          Established
  Encapsulation:  vxlan
  Vlan:           13      <<— VLAN Layer 2 VNI
    Ethernet-Tag: 0
    State:        Established
    Core If:      Vlan99     <<— Interface handling IP VRF forwarding
    Access If:    Vlan13
    NVE If:       nve1
    RMAC:         7486.0bc4.b75d   <<— RMAC is the BIA of SVI 99 Core interface
    Core Vlan:    99
    L2 VNI:       13013
    L3 VNI:       99999
    VTEP IP:      10.255.2.1   <<— Local Tunnel endpoint IP address
    MCAST IP:     239.13.13.13
    VRF:          vxlan     <<— IP VRF for Layer 3 VPN
    Pseudoports:
      GigabitEthernet2/0/1 service instance 13
```

Verify that the Remote Layer 3 VNI Details are Learned on Each VTEP

The following examples show how to verify that the remote Layer 3 VNI details are learned on each VTEP:

```
VTEP-1# show nve peers
Interface  VNI      Type Peer-IP          RMAC/Num_RTs   eVNI     state flags UP time
nve1       99999    L3CP 10.255.2.1       7486.0bc4.b75d 99999      UP   A/M 1w1d
   <<— Layer 3 Control Plane (L3CP), RMAC of Remote VTEP and Uptime of peer are displayed

VTEP-2# show nve peers
Interface  VNI      Type Peer-IP          RMAC/Num_RTs   eVNI     state flags UP time
nve1       99999    L3CP 10.255.1.1       7035.0956.7edd 99999      UP   A/M 21:27:36
   <<— Layer 3 Control Plane (L3CP), RMAC of Remote VTEP and Uptime of peer are displayed
```

Verify that the Layer 3 VNI Tunnel Pseudoport is Installed into Layer 2 FIB in the Core VLAN

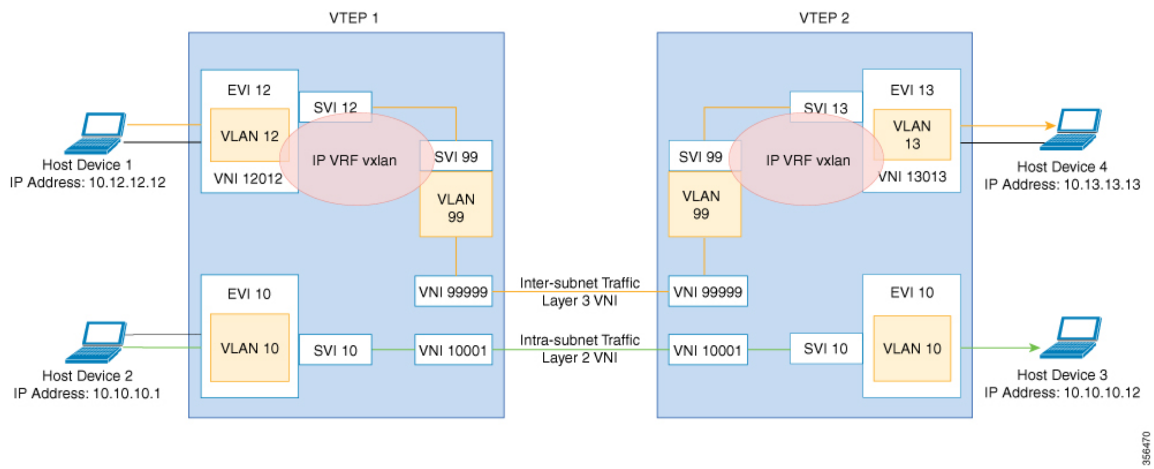
The following examples show how to verify that the Layer 3 VNI tunnel pseudoport is installed into Layer 2 FIB in the core VLAN:

```
VTEP-1# show l2fib bridge-domain 99 detail
   <<— The Core VLAN can be obtained in the output of the
       show l2vpn evpn evi <evpn-instance> detail command
Bridge Domain : 99
  Reference Count : 8
  Replication ports count : 0
  Unicast Address table size : 1
  IP Multicast Prefix table size : 3

  Flood List Information :
    Olist: 5112, Ports: 0
```

```
  VxLAN Information :

  Unicast Address table information :
    7486.0bc4.b75d  VXLAN_CP  L:99999:10.255.1.1 R:99999:10.255.2.1
   <<— Encapsulation Information to reach remote VTEP-2

  IP Multicast Prefix table information :
    Source: *, Group: 224.0.0.0/24, IIF: Null, Adjacency: Olist: 5112, Ports: 0
    Source: *, Group: 224.0.1.39, IIF: Null, Adjacency: Olist: 5112, Ports: 0
    Source: *, Group: 224.0.1.40, IIF: Null, Adjacency: Olist: 5112, Ports: 0


VTEP-2# show l2fib bridge-domain 99 detail
   <<— The Core VLAN can be obtained in the output of the
       show l2vpn evpn evi <evpn-instance> detail command

Bridge Domain : 99
  Reference Count : 8
  Replication ports count : 0
  Unicast Address table size : 1
  IP Multicast Prefix table size : 3

  Flood List Information :
    Olist: 5111, Ports: 0

  VxLAN Information :

  Unicast Address table information :
    7035.0956.7edd  VXLAN_CP  L:99999:10.255.2.1 R:99999:10.255.1.1
   <<— Encapsulation Information to reach remote VTEP-2

  IP Multicast Prefix table information :
    Source: *, Group: 224.0.0.0/24, IIF: Null, Adjacency: Olist: 5111, Ports: 0
    Source: *, Group: 224.0.1.39, IIF: Null, Adjacency: Olist: 5111, Ports: 0
    Source: *, Group: 224.0.1.40, IIF: Null, Adjacency: Olist: 5111, Ports: 0
```

Verifying Inter-Subnet Traffic Movement and Symmetric IRB in an EVPN VXLAN Layer 3 Overlay Network

The following figure illustrates the movement of traffic from host devices connected to VTEP 1 to host devices connected to VTEP 2:



In the above figure, Layer 3 traffic moves from host device 1 to host device 4 through the Layer 3 VNI 99999. To verify the movement of inter-subnet traffic in the EVPN VXLAN Layer 3 overlay network, perform these checks:

1. Verify that Local MAC Address and IP Address Entries are Learned in SISF Device Tracking Table, on page 151
2. Verify that MAC Address and IP Address Entries are Learned in EVPN Manager, on page 152
3. Verify that MAC Address and IP Address Entries are Learned in Layer 2 RIB, on page 153
4. Verify that Local MAC Address and IP Address Entries are Learned in MAC VRF, on page 153
5. Verify that Remote MAC-IP Address Pair is Learend in the VRF, on page 154
6. Verify that IP Routes are Inserted in RIB, on page 155
7. Verify that the Adjacency Table Contains Entries for the VRF-Enabled Core VLAN Interface, on page 155
8. Confirm that Adjacency Exists to the VTEP Tunnel IP Address for a Host Device in IP VRF, on page 156
9. Confirm that Adjacency Exists to Reach Tunnel Destination, on page 156
10. Confirm that the ICMP Echo Request that Leaves Encapsulated from the Source VTEP Reaches the Loopback Tunnel Endpoint and UDP Destination Port on the Destination VTEP Through the Layer 3 VNI and IP VRF, on page 156

Verify that Local MAC Address and IP Address Entries are Learned in SISF Device Tracking Table

The following examples show how to verify that local MAC address and IP address entries are learned in SISF device tracking table:

```
VTEP-1# show device-tracking database vlanid 12
Binding Table has 4 entries, 2 dynamic (limit 100000)
Codes: L - Local, S - Static, ND - Neighbor Discovery, ARP - Address Resolution Protocol,
DH4 - IPv4 DHCP, DH6 - IPv6 DHCP, PKT - Other Packet, API - API created
Preflevel flags (prlvl):
0001:MAC and LLA match     0002:Orig trunk            0004:Orig access
```

```
0008:Orig trusted trunk    0010:Orig trusted access   0020:DHCP assigned
0040:Cga authenticated     0080:Cert authenticated    0100:Statically assigned


    Network Layer Address              Link Layer Address Interface        vlan prlvl  age
   state     Time left
ARP 10.12.12.12                            005f.8602.10e7  Tw1/0/1          12  0005  115s
  REACHABLE  N/A

VTEP-2# show device-tracking database vlanid 13
vlanDB has 2 entries for vlan 13, 1 dynamic
Codes: L - Local, S - Static, ND - Neighbor Discovery, ARP - Address Resolution Protocol,
DH4 - IPv4 DHCP, DH6 - IPv6 DHCP, PKT - Other Packet, API - API created
Preflevel flags (prlvl):
0001:MAC and LLA match     0002:Orig trunk            0004:Orig access
0008:Orig trusted trunk    0010:Orig trusted access   0020:DHCP assigned
0040:Cga authenticated     0080:Cert authenticated    0100:Statically assigned


    Network Layer Address              Link Layer Address Interface        vlan prlvl  age
   state     Time left
ARP 10.13.13.13                            008e.7391.1977  Gi2/0/1          13  0005  155s
  REACHABLE  N/A
```

Verify that MAC Address and IP Address Entries are Learned in EVPN Manager

The following examples show how to verify that MAC address and IP address entries are learned in EVPN manager:

```
VTEP-1# show l2vpn evpn mac ip evi 12
IP Address                EVI   VLAN  MAC Address    Next Hop
------------------------- ----- ----- -------------- -------------------------
10.12.12.12               12    12    005f.8602.10e7 Tw1/0/1:12

VTEP-1#sh l2vpn evpn mac ip evi 12 detail
IP Address:                10.12.12.12
EVPN Instance:             12
Vlan:                      12
MAC Address:               005f.8602.10e7
Ethernet Segment:          0000.0000.0000.0000.0000
Ethernet Tag ID:           0
Next Hop:                  TwoGigabitEthernet1/0/1 service instance 12
VNI:                       12012
Sequence Number:           0
IP Duplication Detection:  Timer not running


VTEP-2# show l2vpn evpn mac ip evi 13
IP Address                EVI   VLAN  MAC Address    Next Hop
------------------------- ----- ----- -------------- -------------------------
10.13.13.13               13    13    008e.7391.1977 Gi2/0/1:13

VTEP-2#sh l2vpn evpn mac ip evi 13 detail
IP Address:                10.13.13.13
EVPN Instance:             13
Vlan:                      13
MAC Address:               008e.7391.1977
Ethernet Segment:          0000.0000.0000.0000.0000
Ethernet Tag ID:           0
Next Hop:                  GigabitEthernet2/0/1 service instance 13
```

```
VNI:                        13013
Sequence Number:            0
IP Duplication Detection:   Timer not running
```

Verify that MAC Address and IP Address Entries are Learned in Layer 2 RIB

The following examples show how to verify that MAC address and IP address entries are learned in Layer 2 RIB:

```
VTEP-1# show l2route evpn mac ip
  EVI       ETag  Prod    Mac Address         Host IP                Next Hop(s)
----- ---------- ----- -------------- --------------- --------------------------
   12          0 L2VPN 005f.8602.10e7     10.12.12.12                Tw1/0/1:12

VTEP-2# show l2route evpn mac ip
  EVI       ETag  Prod    Mac Address         Host IP                Next Hop(s)
----- ---------- ----- -------------- --------------- --------------------------
   13          0 L2VPN 008e.7391.1977     10.13.13.13                Gi2/0/1:13
```

Verify that Local MAC Address and IP Address Entries are Learned in MAC VRF

```
VTEP-1# show bgp l2vpn evpn evi 12 route-type 2 0 005F860210E7 10.12.12.12
BGP routing table entry for [2][10.1.1.1:12][0][48][005F860210E7][32][10.12.12.12]/24,
version 72
Paths: (1 available, best #1, table evi_12)     <<— The Layer 2 VPN table number
                                                    for EVPN instance 12
  Advertised to update-groups:
     1
  Refresh Epoch 1
  Local     <<— Indicates locally learned route
    :: (via default) from 0.0.0.0 (10.1.1.1)
      Origin incomplete, localpref 100, weight 32768, valid, sourced, local, best
      EVPN ESI: 00000000000000000000, Label1 12012, Label2 99999 <<— Displays both Layer 2
                                                                     and VRF labels
      Extended Community: RT:12:1 RT:10.255.1.1:1 ENCAP:8    <<— Note the VRF stitching RT
                                                                 as well as the Layer 2 RT
        Router MAC:7035.0956.7EDD
      Local irb vxlan vtep:
        vrf:vxlan, l3-vni:99999
        local router mac:7035.0956.7EDD   <<— Local RMAC
        core-irb interface:Vlan99    <<— VRF Layer 3 VPN interface
        vtep-ip:10.255.1.1    <<— Loopback 999 tunnel endpoint
      rx pathid: 0, tx pathid: 0x0
```

The following examples show how to verify that local MAC address and IP address entries are learned in MAC VRF:

```
VTEP-2# show bgp l2vpn evpn evi 13 route-type 2 0 008E73911977 10.13.13.13
BGP routing table entry for [2][10.2.2.2:13][0][48][008E73911977][32][10.13.13.13]/24,
version 70
Paths: (1 available, best #1, table evi_13)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  Local     <<— Indicates locally learned route
    :: (via default) from 0.0.0.0 (10.2.2.2)
      Origin incomplete, localpref 100, weight 32768, valid, sourced, local, best
```

```
      EVPN ESI: 00000000000000000000, Label1 13013, Label2 99999
      Extended Community: RT:13:1 RT:10.255.2.1:1 ENCAP:8
        Router MAC:7486.0BC4.B75D
      Local irb vxlan vtep:
        vrf:vxlan, l3-vni:99999
        local router mac:7486.0BC4.B75D
        core-irb interface:Vlan99
        vtep-ip:10.255.2.1
      rx pathid: 0, tx pathid: 0x0
```

Verify that Remote MAC-IP Address Pair is Learend in the VRF

The following examples verify that remote MAC-IP address pair is learned in the VRF:

```
VTEP-1# show bgp vpnv4 unicast vrf vxlan 10.13.13.13
BGP routing table entry for 10.255.1.1:1:10.13.13.13/32, version 15
Paths: (1 available, best #1, table vxlan)         <<— VPNv4 VRF BGP table
  Not advertised to any peer
  Refresh Epoch 2
  Local, imported path from [2][10.2.2.2:13][0][48][008E73911977][32][10.13.13.13]/24
(global)
   <<— EVPN type-2, l2vpn RD 10.2.2.2:13, MAC and IP addresses
    10.255.2.1 (metric 3) (via default) from 10.2.2.2 (10.2.2.2)
   <<— Next hop 10.255.2.1, learned from RR 10.2.2.2
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: ENCAP:8 Router MAC:7486.0BC4.B75D
      Originator: 10.2.2.2, Cluster list: 10.2.2.2
      Local vxlan vtep:
        vrf:vxlan, vni:99999
        local router mac:7035.0956.7EDD
        encap:8
        vtep-ip:10.255.1.1
        bdi:Vlan99
      Remote VxLAN:
        Topoid 0x2(vrf vxlan)    <<— VRF vxlan (mapped to ID 2)
        Remote Router MAC:7486.0BC4.B75D  <<— VTEP-2 RMAC
        Encap 8     <<— VXLAN encap (type 8)
        Egress VNI 99999    <<— VRF VNI
        RTEP 10.255.2.1     <<— VTEP-2 Remote Tunnel Endpoint
      rx pathid: 0, tx pathid: 0x0

VTEP-2# show bgp vpnv4 unicast vrf vxlan 10.12.12.12
BGP routing table entry for 10.255.2.1:1:10.12.12.12/32, version 15
Paths: (1 available, best #1, table vxlan)
  Not advertised to any peer
  Refresh Epoch 2
  Local, imported path from [2][10.1.1.1:12][0][48][005F860210E7][32][10.12.12.12]/24
(global)
   <<— EVPN type-2, l2vpn RD 10.1.1.1:12, MAC and IP addresses
    10.255.1.1 (metric 3) (via default) from 10.2.2.2 (10.2.2.2)
   <<— Next hop 10.255.1.1, learned from RR 10.2.2.2
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: ENCAP:8 Router MAC:7035.0956.7EDD
      Originator: 10.1.1.1, Cluster list: 10.2.2.2
      Local vxlan vtep:
        vrf:vxlan, vni:99999
        local router mac:7486.0BC4.B75D
        encap:8
        vtep-ip:10.255.2.1
        bdi:Vlan99
      Remote VxLAN:
```

```
         Topoid 0x2(vrf vxlan)     <<— VRF vxlan (mapped to ID 2)
         Remote Router MAC:7035.0956.7EDD  <<— VTEP-1 RMAC
         Encap 8     <<— VXLAN encap (type 8)
         Egress VNI 99999     <<— VRF VNI
         RTEP 10.255.1.1     <<— VTEP-2 Remote Tunnel Endpoint
       rx pathid: 0, tx pathid: 0x0
```

Verify that IP Routes are Inserted in RIB

The following examples show how to verify that IP routes are inserted in RIB:

```
VTEP-1# show ip route vrf vxlan 10.13.13.13

Routing Table: vxlan
Routing entry for 10.13.13.13/32
  Known via "bgp 69420", distance 200, metric 0, type internal
  Last update from 10.255.2.1 on Vlan99, 00:11:33 ago
  Routing Descriptor Blocks:
  * 10.255.2.1 (default), from 10.2.2.2, 00:11:33 ago, via Vlan99 <<— Next hop here is the
                                                                         Core VLAN interface
      Route metric is 0, traffic share count is 1
      AS Hops 0
      MPLS label: none


VTEP-2# show ip route vrf vxlan 10.12.12.12

Routing Table: vxlan
Routing entry for 10.12.12.12/32
  Known via "bgp 69420", distance 200, metric 0, type internal
  Last update from 10.255.1.1 on Vlan99, 00:04:06 ago
  Routing Descriptor Blocks:
  * 10.255.1.1 (default), from 10.2.2.2, 00:04:06 ago, via Vlan99 <<— Next hop here is the
                                                                         Core VLAN interface
      Route metric is 0, traffic share count is 1
      AS Hops 0
      MPLS label: none
```

Verify that the Adjacency Table Contains Entries for the VRF-Enabled Core VLAN Interface

The following examples show how to verify that the adjacency table contains entries for the VRF-enabled core VLAN interface:

```
VTEP-1# show adjacency vlan 99 detail
Protocol Interface                 Address
IP       Vlan99                    10.255.2.1(9)    <<— IP unnumbered from Loopback 999
                                   0 packets, 0 bytes
                                   epoch 0
                                   sourced in sev-epoch 6
                                   Encap length 14
                                   74860BC4B75D703509567EDD0800
   <<— Local RMAC is 74860BC4B75D, Remote RMAC is 703509567EDD, etype is 800
                                   VXLAN Transport tunnel
   <<— Tunnel Interface (RMAC, using VTEP Loopback IP address)

VTEP-2# show adjacency vlan 99 detail
Protocol Interface                 Address
IP       Vlan99                    10.255.1.1(9)    <<— IP unnumbered from Loopback 999
```

```
                                   0 packets, 0 bytes
                                   epoch 0
                                   sourced in sev-epoch 5
                                   Encap length 14
                                   703509567EDD74860BC4B75D0800
<<— Local RMAC is 703509567EDD, Remote RMAC is 74860BC4B75D, etype is 800
                                   VXLAN Transport tunnel
<<— Tunnel Interface (RMAC, using VTEP Loopback IP address)
```

Confirm that Adjacency Exists to the VTEP Tunnel IP Address for a Host Device in IP VRF

The following example shows how to confirm that adjacency exists to the VTEP Tunnel IP address for a host device in IP VRF:

```
VTEP-1# show ip cef vrf vxlan 10.13.13.13/32  <<— Remote host in VLAN 13 of VTEP-2
10.13.13.13/32
  nexthop 10.255.2.1 Vlan99
```

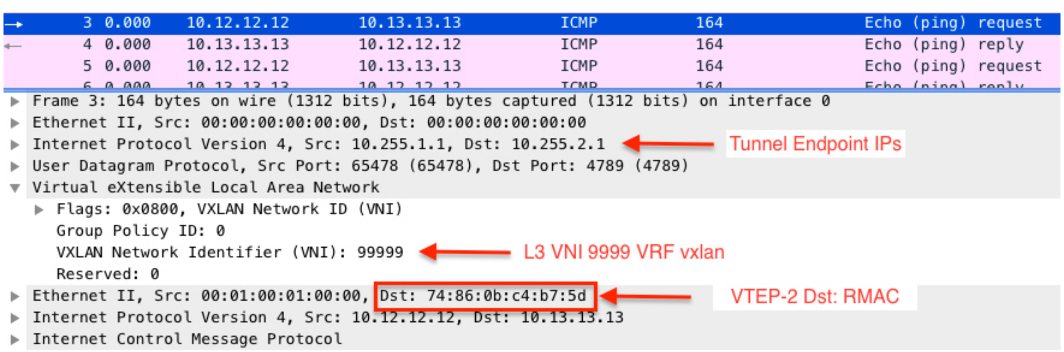
Confirm that Adjacency Exists to Reach Tunnel Destination

The following example shows how to confirm that adjacency exists to reach tunnel destination:

```
VTEP-1# show ip cef 10.255.1.11
10.255.2.1/32
  nexthop 10.1.1.6 TwoGigabitEthernet1/0/2
```

Confirm that the ICMP Echo Request that Leaves Encapsulated from the Source VTEP Reaches the Loopback Tunnel Endpoint and UDP Destination Port on the Destination VTEP Through the Layer 3 VNI and IP VRF

The following image confirms that the ICMP echo request that leaves encapsulated from source VTEP reaches the Loopback interface and UDP destination port on the destination VTEP through the Layer 3 VNI and IP VRF:



Troubleshooting Unicast Forwarding Between a VXLAN Network and an IP Network

This scenario might occur when host device 1 attempts to ping an external IP address through a border leaf VTEP. Perform the checks listed in the following table before troubleshooting unicast forwarding between a VXLAN network and an external IP network.

Table 15: Scenario 4: Troubleshooting Unicast Forwarding Between a VXLAN Network and an IP Network

Check to be performed	Steps to follow
Is one IP address present in the VXLAN network and the other IP address coming from external IP network?	Check the local subnets (or the SVI interfaces) if the remote subnet is present. **Note** Local subnet has the remote subnet listed even in the case of scenario 3.
Is the EVPN route type 5 being used to send traffic to remote destination?	Run the **show bgp l2vpn evpn all** command in privileged EXEC mode on the VTEP. Look for remote prefix to be displayed as `[5]` for route type 5.

To troubleshoot unicast forwarding between a VXLAN network and an external IP network, follow these steps:

- Verify the provisioning of the EVPN VXLAN Layer 3 overlay network.
- Verify traffic movement from the VXLAN network to the IP network through the border leaf switch using route type 5.

Verifying the Provisioning of an EVPN VXLAN Layer 3 Overlay Network

See Verifying the Provisioning of an EVPN VXLAN Layer 3 Overlay Network, on page 145 for detailed steps.

Verifying Traffic from a VXLAN Fabric to an IP Network Through a Border Leaf Switch Using Route Type 5

To verify the movement of traffic from a VXLAN fabric to an external IP network through a border leaf switch, perform these checks:

1. Check the Table Entries for BGP, EVPN, and VPNv4 Tables, on page 157
2. Check the Table Entries for BGP, EVPN, and VPNv4 Tables, on page 157
3. Confirm that Adjacency exists to Reach Tunnel Destination, on page 160

Check the Table Entries for BGP, EVPN, and VPNv4 Tables

The following examples show how to check the table entries for BGP, EVPN and VPNv4 tables:

```
VTEP-1# show bgp vpnv4 unicast vrf vxlan 10.9.9.9/32
   <<— To a remote IP address outside the VXLAN fabric
BGP routing table entry for 10.255.1.1:1:10.9.9.9/32, version 150
Paths: (1 available, best #1, table vxlan)     <<— VPNv4 VRF BGP table
  Not advertised to any peer
  Refresh Epoch 2
  Local, imported path from [5][10.255.1.11:1][0][32][10.9.9.9]/17 (global)
   <<— Learned from EVPN into VPNv4
    10.255.1.11 (metric 3) (via default) from 10.2.2.2 (10.2.2.2)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Extended Community: ENCAP:8 Router MAC:EC1D.8B55.F55D
      Originator: 10.255.1.11, Cluster list: 10.2.2.2
      Local vxlan vtep:
        vrf:vxlan, vni:99999
        local router mac:7035.0956.7EDD
        encap:8
        vtep-ip:10.255.1.1
        bdi:Vlan99
      Remote VxLAN:
        Topoid 0x2(vrf vxlan)
        Remote Router MAC:EC1D.8B55.F55D  <<— Border_Leaf_VTEP RMAC
        Encap 8
        Egress VNI 99999      <<— VNI associated with VRF
        RTEP 10.255.1.11      <<— Tunnel IP address
      rx pathid: 0, tx pathid: 0x0


VTEP-1# show bgp l2vpn evpn all route-type 5 0 10.9.9.9 32
   <<— This is sent as type 5 as there is no VNI at all for it to be mapped to
BGP routing table entry for [5][10.255.1.11:1][0][32][10.9.9.9]/17, version 650
Paths: (1 available, best #1, table EVPN-BGP-Table)
  Not advertised to any peer
  Refresh Epoch 2
  Local
    10.255.1.11 (metric 3) (via default) from 10.2.2.2 (10.2.2.2)
   <<— Border_Leaf_VTEP Tunnel IP address
      Origin IGP, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Gateway Address: 0.0.0.0, VNI Label 99999, MPLS VPN
Label 0
   <<— Using Layer 3 VNI 99999
      Extended Community: RT:10.255.1.11:1 ENCAP:8 Router MAC:EC1D.8B55.F55D
   <<— Route Target and RMAC of Border_Leaf_VTEP
      Originator: 10.255.1.11, Cluster list: 10.2.2.2
      rx pathid: 0, tx pathid: 0x0


Border_Leaf_VTEP# show bgp vpnv4 unicast vrf vxlan 10.12.12.12/32
   <<— To VXLAN Fabric IP address on VTEP-1
BGP routing table entry for 10.255.1.11:1:10.12.12.12/32, version 3092
Paths: (1 available, best #1, table vxlan)
Not advertised to any peer
Refresh Epoch 4
Local, imported path from [2][10.1.1.1:12][0][48][005F860210E7][32][10.12.12.12]/24 (global)

   <<— EVPN type-2 has been imported to VPNv4, from VTEP-1
  10.255.1.1 (metric 3) (via default) from 10.2.2.2 (10.2.2.2)
    Origin incomplete, metric 0, localpref 100, valid, internal, best
    Extended Community: RT:10.255.1.11:1 ENCAP:8 Router MAC:7035.0956.7EDD
    Originator: 10.1.1.1, Cluster list: 10.2.2.2
    Local vxlan vtep:
      vrf:vxlan, vni:99999
      local router mac:EC1D.8B55.F55D
      encap:8
```

```
      vtep-ip:10.255.1.11
      bdi:Vlan99
    Remote VxLAN:
      Topoid 0x2(vrf vxlan)
      Remote Router MAC:7035.0956.7EDD  <<— VTEP-1 RMAC
       Encap 8
       Egress VNI 99999
       RTEP 10.255.1.1                  <<— VTEP-1 Tunnel IP address
       rx pathid: 0, tx pathid: 0x0

Border_Leaf_VTEP# show bgp l2vpn evpn all route-type 2 0 005F860210E7 10.12.12.12
   <<— Border_Leaf_VTEP still knows the type-2. This is still exchanged between the VTEPs
       even though the prefix has been imported to VPNv4
BGP routing table entry for [2][10.1.1.1:12][0][48][005F860210E7][32][10.12.12.12]/24,
version 3085
Paths: (1 available, best #1, table EVPN-BGP-Table)
Not advertised to any peer
Refresh Epoch 4
Local
10.255.1.1 (metric 3) (via default) from 10.2.2.2 (10.2.2.2)
  Origin incomplete, metric 0, localpref 100, valid, internal, best
  EVPN ESI: 00000000000000000000, Label1 12012, Label2 99999
   <<— Both Layer 2 VNI and Layer 3 VNI labels are seen in type-2,
       but only Layer 3 VNI 99999 is used, once imported to VPNv4
  Extended Community: RT:12:1 RT:10.255.1.1:1 ENCAP:8
    Router MAC:7035.0956.7EDD
  Originator: 10.1.1.1, Cluster list: 10.2.2.2
  rx pathid: 0, tx pathid: 0x0
```

Note To check if IP routes have been inserted into CEF table, run the **show ip route vrf** *vrf-name* command in privileged EXEC mode.

Confirm that Adjacency Exists to the VTEP Tunnel IP Address for the Host Device in IP VRF

The following examples show how to confirm that adjacency exists to the VTEP Tunnel IP address for the host device in IP VRF:

```
VTEP-1# show ip cef vrf vxlan 10.9.9.9/32 platform
10.9.9.9/32
  Platform adj-id: 0x1A, 0x0, tun_qos_dpidx:0   <<— Adjacency ID to remote IP address

VTEP-1# show platform software fed sw ac matm macTable vlan 99
VLAN   MAC                   Type  Seq#   EC_Bi  Flags  machandle           siHandle
      riHandle             diHandle              *a_time  *e_time  ports
------------------------------------------------------------------------------------------
99     7035.0956.7edd      0x8002       0       0      64   0x7ffa48d61be8      0x7ffa48d630b8
      0x0                 0x5154                        0         0  Vlan99
99     7486.0bc4.b75d    0x1000001       0       0      64   0x7ffa48fb1bb8      0x7ffa48fac698
      0x7ffa48fab038      0x7ffa4838cc18                0         0  RLOC 10.255.2.1 adj_id
103
99     ec1d.8b55.f55d    0x1000001       0       0      64   0x7ffa48d065e8      0x7ffa48d01d08
      0x7ffa48c9a618      0x7ffa4838cc18                0         0  RLOC 10.255.1.11 adj_id
47
```

Confirm that Adjacency exists to Reach Tunnel Destination

The following example shows how to confirm that adjacency exists to reach Tunnel destination:

```
VTEP-1# show ip cef 10.255.1.11
10.255.1.11/32
  nexthop 10.1.1.6 TwoGigabitEthernet1/0/2
```

CHAPTER 8

Feature History for BGP EVPN VXLAN

Feature History for BGP EVPN VXLAN

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.12.1	EVPN VXLAN Overlay Network for IPv4 Bridged Traffic	EVPN VXLAN overlay network for IPv4 bridged traffic is a Layer 2 overlay network that allows host devices within the same subnet to send IPv4 bridged traffic to each other using a Layer 2 virtual network instance (VNI).
	EVPN VXLAN Overlay Network for IPv4 Routed Traffic	EVPN VXLAN overlay network for IPv4 routed traffic is a Layer 3 overlay network that allows host devices in different Layer 2 networks to send IPv4 routed traffic to each other using a Layer 3 VNI and an IP VRF.
	Layer 2 Broadcast, Unknown Unicast, and Multicast (BUM) Traffic Forwarding using Underlay Multicast	Multi-destination Layer 2 broadcast, unknown unicast, and multicast (BUM) traffic in an EVPN VXLAN network is replicated through a multicast group in the underlay network and forwarded to all the endpoints of the network.
	Leaf Functionality	A leaf switch sits on the edge of a BGP EVPN VXLAN fabric and is connected to the host or access devices. It functions as a virtual tunnel end point (VTEP) and performs encapsulation and decapsulation.
	EVPN VXLAN Integrated and Bridging	EVPN VXLAN integrated and bridging (IRB) allows the VTEPs in a VXLAN network to forward both Layer 2 or bridged traffic and Layer 3 or routed traffic. It is implemented as symmetric and asymmetric IRB.
	EVPN VXLAN Distributed Anycast Gateway	EVPN VXLAN distributed anycast gateway is a default gateway addressing mechanism that enables the use of the same gateway IP address across all the leaf switches that are part of a VXLAN network. Support was introduced with manual MAC address configuration on the Layer 2 VNI VLAN's switch virtual interface (SVI) on all VTEPs as the only method to enable the feature.
	DHCP Relay for IPv4 Traffic in BGP EVPN VXLAN Fabric	The VTEP in a BGP EVPN VXLAN fabric is configured as a DHCP relay agent to provide DCHP relay services for IPv4 traffic in a multi-tenant VXLAN environment.
	EVPN VXLAN Overlay Network for IPv6 Bridged Traffic	EVPN VXLAN overlay network for IPv6 bridged traffic is a Layer 2 overlay network that allows host devices within the same subnet to send IPv6 bridged traffic to each other using a Layer 2 VNI.
	EVPN VXLAN Overlay Network for IPv6 Routed Traffic	EVPN VXLAN overlay network for IPv6 routed traffic is a Layer 3 overlay network that allows host devices in different Layer 2 networks to send IPv6 routed traffic to each other using a Layer 3 VNI and an IP VRF.

Release	Feature	Feature Information
	Layer 2 Broadcast, Unknown Unicast, and Multicast (BUM) Traffic Forwarding using Ingress Replication	Ingress replication is a unicast approach to handle multi-destination Layer 2 BUM traffic in an EVPN VXLAN network. It involves an ingress device replicating every incoming BUM packet and sending them as a separate unicast to the remote egress devices.
	MAC Aliasing for EVPN VXLAN Distributed Anycast Gateway	MAC aliasing allows the leaf switches in an EVPN VXLAN network to advertise the MAC addresses of their Layer 2 VLAN's SVI as the gateway MAC address to all the other leaf switches in the network. MAC aliasing removes the need to explicitly configure the same MAC address on the Layer 2 VNI VLAN's SVI on all VTEPs in order to enable distributed anycast gateway.
	EVPN VXLAN Multihoming in Single-active Redundancy Mode	Multi-homing provides redundancy in the connection between a customer edge (CE) device and a VTEP by connecting the customer network with multiple VTEPs in an EVPN VXLAN network. In single-active redundancy mode, only one VTEP, among a group of VTEPs that are attached to the particular ethernet segment, is allowed to forward traffic to and from that ethernet segment. Multi-homing in single-active redundancy mode was introduced only in the form of dual-homing, allowing a CE device to be connected to two VTEPs. This feature was introduced.
	Border Leaf Functionality	A border leaf switch is a leaf switch in a BGP EVPN VXLAN fabric that enables external connectivity with other Layer 2 and Layer 3 networks by acting as the connecting node between the two networks.
	Autonomous System Number Rewrite	The **rewrite-evpn-rt-asn** command was introduced to enable the rewrite of the autonomous system number (ASN) portion of the EVPN route target that originates from the current autonomous system with the ASN of the target eBGP EVPN peer.
	VRF-Lite Border Leaf Handoff	VRF-Lite border leaf handoff in a BGP EVPN VXLAN fabric allows Layer 3 external connectivity with a VRF-Lite network through a border leaf switch.
	MPLS Layer 3 VPN Border Leaf Handoff	MPLS Layer 3 VPN border leaf handoff in a BGP EVPN VXLAN fabric allows Layer 3 external connectivity with an MPLS Layer 3 VPN network through a border leaf switch.
	IEEE 802.1Q Border Leaf Handoff	

Release	Feature	Feature Information
		IEEE 802.1Q border leaf handoff in a BGP EVPN VXLAN fabric allows Layer 2 external connectivity with an IEEE 802.1Q network through a border leaf switch.
	Access Border Leaf Handoff	Access border leaf handoff in a BGP EVPN VXLAN fabric allows Layer 2 external connectivity with an Access network through a border leaf switch.
	VPLS over MPLS Border Leaf Handoff	VPLS over MPLS border leaf handoff in a BGP EVPN VXLAN fabric allows Layer 2 external connectivity with a VPLS over MPLS network through a border leaf switch.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.