



## Troubleshooting BGP EVPN VXLAN

- [Troubleshooting Scenarios for BGP EVPN VXLAN, on page 1](#)
- [Troubleshooting Broadcast, Unknown Unicast, Multicast Traffic Forwarding, on page 2](#)
- [Troubleshooting Unicast Forwarding Between VTEPs in the Same VLAN Through a Layer 2 VNI, on page 6](#)
- [Troubleshooting Unicast Forwarding Between VTEPs in Different VLANs Through a Layer 3 VNI, on page 18](#)
- [Troubleshooting Unicast Forwarding Between a VXLAN Network and an IP Network, on page 31](#)

## Troubleshooting Scenarios for BGP EVPN VXLAN

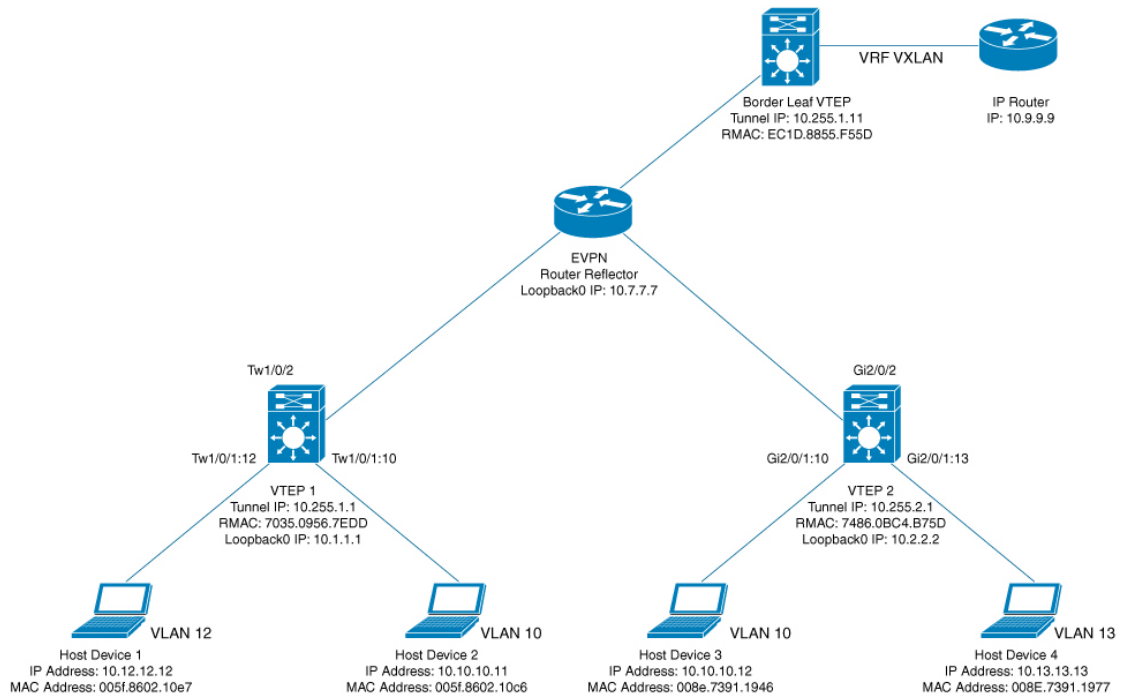
This document provides information about the various troubleshooting scenarios that are applicable to BGP EVPN VXLAN and how to troubleshoot each scenario.

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 precedes 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 1: EVPN VXLAN Topology



The following are the various troubleshooting scenarios that apply to BGP EVPN VXLAN for the topology illustrated in the [Figure 1: 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 1: EVPN VXLAN Topology, on page 2](#). Perform the checks listed in the following table before troubleshooting BUM traffic forwarding:

**Table 1: Scenario 1: Broadcast, Unknown 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: <ul style="list-style-type: none"> <li>• Check the host device.</li> <li>• Check the SVI configuration on the VTEP.</li> </ul>   |
| Has the remote MAC address been learned for unknown unicast traffic? | Run the <b>show platform software fed switch active macTable vlan <i>vlan-id</i></b> command in privileged EXEC mode on the local VTEP and check if the MAC address of the remote host device is displayed 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 3](#)
2. [Set Up Embedded Capture Towards the Core-Facing Interface, on page 4](#)
3. [Ping the Remote Host Device, on page 4](#)
4. [Verify that an ARP Request Has Been Received and a Multicast Route Has Been Built, on page 4](#)
5. [Confirm the Presence of ARP Request Replies in Embedded Capture, on page 5](#)
6. [Verify that the Encapsulated ARP Request is Leaving in a Multicast Group to a VXLAN UDP Destination Port, on page 5](#)
7. [Verify that the ARP Reply from Core Interface is Encapsulated in Unicast to a VXLAN UDP Destination Port, on page 6](#)

**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.

| No. | Time  | Source            | Destination       | Protocol | Length | Info                                  |
|-----|-------|-------------------|-------------------|----------|--------|---------------------------------------|
| 1   | 0.000 | 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 |
| 2   | 0.000 | 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   |

```

> Frame 1: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interface 0
v Ethernet II, Src: 74:a2:e6:4f:c9:00, Dst: 01:00:5e:0a:0a:0a
  > Destination: 01:00:5e:0a:0a:0a
  > Source: 74:a2:e6:4f:c9:00
  Type: IPv4 (0x0800)
  > Internet Protocol Version 4, Src: 10.255.1.1, Dst: 239.10.10.10
  v User Datagram Protocol, Src Port: 65419 (65419), Dst Port: 4789 (4789)
    Source Port: 65419
    Destination Port: 4789
    Length: 76
    > Checksum: 0x0000 (none)
    [Stream index: 0]
  v Virtual eXtensible Local Area Network
    > Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 10001
    Reserved: 0
  v Ethernet II, Src: 00:5f:86:02:10:c6, Dst: ff:ff:ff:ff:ff:ff
    > Destination: ff:ff:ff:ff:ff:ff
    > Source: 00:5f:86:02:10:c6
    Type: ARP (0x0806)
    Trailer: 0000000000000000000000000000000000000000000000000000000000000000
    Address Resolution Protocol (request)
  
```

**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.

| No. | Time  | Source            | Destination       | Protocol | Length | Info                                  |
|-----|-------|-------------------|-------------------|----------|--------|---------------------------------------|
| 1   | 0.000 | 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 |
| 2   | 0.000 | 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   |

```

> Frame 2: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interface 0
v Ethernet II, Src: 74:a2:e6:4f:c9:00, Dst: 70:35:09:56:7e:d6
  > Destination: 70:35:09:56:7e:d6
  > Source: 74:a2:e6:4f:c9:00
  Type: IPv4 (0x0800)
  > Internet Protocol Version 4, Src: 10.255.1.2, Dst: 10.255.1.1
  v User Datagram Protocol, Src Port: 65350 (65350), Dst Port: 4789 (4789)
    Source Port: 65350
    Destination Port: 4789
    Length: 76
    > Checksum: 0x0000 (none)
    [Stream index: 1]
  v Virtual eXtensible Local Area Network
    > Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 10001
    Reserved: 0
  v Ethernet II, Src: 28:52:61:bf:a9:46, Dst: 00:5f:86:02:10:c6
    > Destination: 00:5f:86:02:10:c6
    > Source: 28:52:61:bf:a9:46
    Type: ARP (0x0806)
    Trailer: 0000000000000000000000000000000000000000000000000000000000000000
    Address Resolution Protocol (reply)
  
```

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 2: 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 <b>arp -a</b> command in privileged EXEC mode on the host device.  |
| Do the hosts have the same subnet masks?   | Perform any of the following steps: <ul style="list-style-type: none"> <li>• Check the host device.</li> <li>• Check the SVI configuration on the VTEP.</li> </ul>   |
| Do you have the EVPN instance configured on your local VTEP?                                 | Run the following commands in privileged EXEC mode on the VTEP: <ul style="list-style-type: none"> <li>• <b>show run   section l2vpn</b></li> <li>• <b>show run   section vlan config</b></li> <li>• <b>show run interface nve interface-number</b></li> </ul> |
| Has the remote MAC address been learned in platform MATM in the same VLAN as the local host? | Run the <b>show platform software fed switch active matm macTable vlan vlan-id</b> 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 7](#)
2. [Ensure that an NVE Peer is Present for the Layer 2 VNI, on page 9](#)
3. [Verify the Provisioning of the Layer 2 VNI in NVE Component, on page 9](#)
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 10](#)

### 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 nvl: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 nvl:10001:239.10.10.10 <<- VXLAN replication group

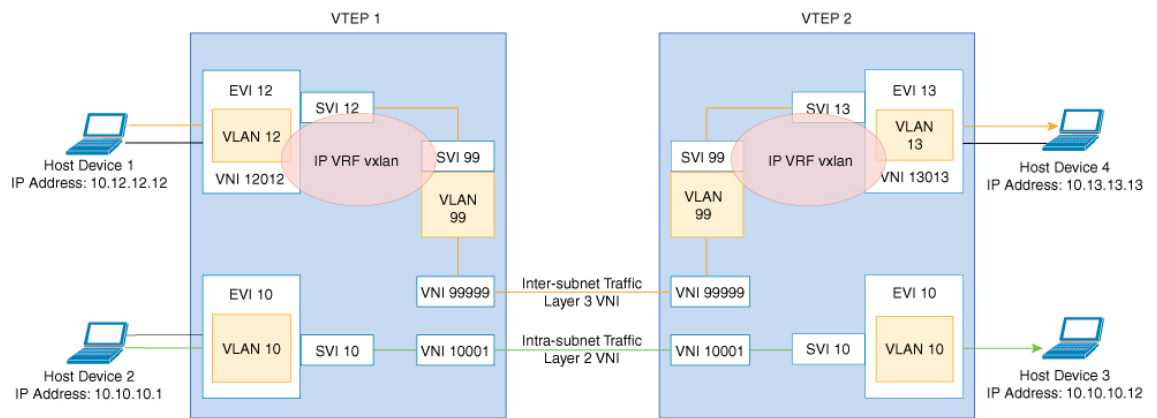
Unicast Address table information :
005f.8602.10c6 VXLAN_CP L:10001:10.255.2.1 R:10001:10.255.1.1

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 2: 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 12](#)
2. [Verify that Both Local and Remote MAC Addresses are Learned in FED-MATM, on page 12](#)

3. Confirm that the ICMP Echo Request Leaves VTEP 1 Encapsulated and Goes to a UDP Destination Port on VTEP 2, on page 13
4. Verify ARP for Local Host Devices, on page 13
5. Verify that the MAC Address Entries are Learned in SISF Device Tracking Table, on page 13
6. Verify that EVPN Manager Has Been Updated with the MAC Address Entries, on page 14
7. Verify that EVPN Manager Has Updated the MAC Routes into Layer 2 RIB, on page 15
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 15
9. Verify that the MAC Routes Learned from BGP and Updated to Layer 2 RIB are Also Updated to L2FIB, on page 17



**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

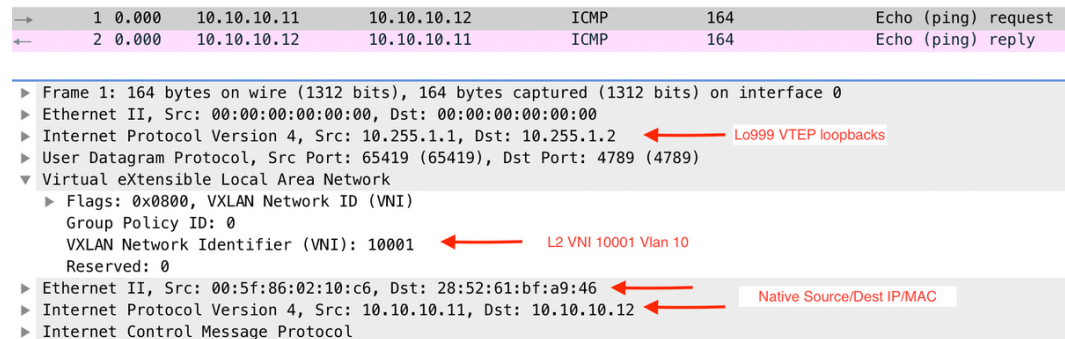
<<- Local MAC address is displayed here

```

**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 3:



**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 12vpn 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 Address 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 12vpn 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 12vpn 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
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 address
EVPN Instance:       10                  <<- EVPN Instance
Vlan:                 10                  <<- VLAN
Ethernet Segment:    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 to 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, 13-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 3: 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 <b>show ip interface brief   exclude unassigned</b> 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: <ul style="list-style-type: none"> <li>• <b>show run   section l2vpn</b></li> <li>• <b>show run   section vlan config</b></li> <li>• <b>show run interface nve interface-number</b></li> </ul> |

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 19](#)
2. [Verify that the IP VRF is Provisioned with the Correct SVIs, Stitching Route-Targets, and Route Distinguisher, on page 20](#)
3. [Verify that Both Layer 2 and Layer 3 VNIs are provisioned in the VRF and are UP, on page 21](#)
4. [Verify that EVPN Manager is Updated from the NVE with all the Layer 2 and IRB Attributes, on page 22](#)
5. [Verify that the Remote Layer 3 VNI Details are Learned on Each VTEP, on page 23](#)
6. [Verify that the Layer 3 VNI Tunnel Pseudoport is Installed into Layer 2 FIB in the Core VLAN, on page 23](#)

### 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
```

```
Tunnell          10.1.1.5          YES unset up          up
nvel             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
Tunnell           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   V110
              V112
              V199
```

```
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:
V110 V112 V199
Address family ipv4 unicast (Table ID = 0x2):    <<- Table 2 maps to VRF vxlan,
                                                also found in BGP 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                V110
                                      V113
                                      V199
    
```

```

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:
V110 V113 V199
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

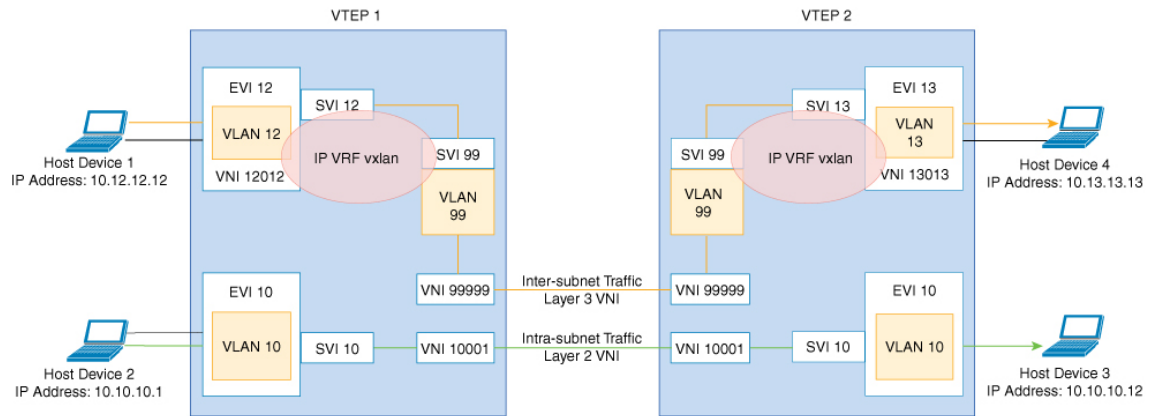
IP Multicast Prefix table information :
 Source: *, Group: 224.0.0.0/24, IIF: Null, Adjacency: Olist: 5111, Ports: 0
 Source: *, Group: 224.0.1.39, IIF: Null, Adjacency: Olist: 5111, Ports: 0
 Source: *, Group: 224.0.1.40, IIF: Null, Adjacency: Olist: 5111, Ports: 0

```

## 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 25
2. Verify that MAC Address and IP Address Entries are Learned in EVPN Manager, on page 26
3. Verify that MAC Address and IP Address Entries are Learned in Layer 2 RIB, on page 27
4. Verify that Local MAC Address and IP Address Entries are Learned in MAC VRF, on page 27
5. Verify that Remote MAC-IP Address Pair is Learend in the VRF, on page 28
6. Verify that IP Routes are Inserted in RIB, on page 29
7. Verify that the Adjacency Table Contains Entries for the VRF-Enabled Core VLAN Interface, on page 29
8. Confirm that Adjacency Exists to the VTEP Tunnel IP Address for a Host Device in IP VRF, on page 30
9. Confirm that Adjacency Exists to Reach Tunnel Destination, on page 30
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 30

**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<br>state | Time left | Link Layer Address | Interface | vlan | prlvl | age  |
|--------------------------------|-----------|--------------------|-----------|------|-------|------|
| 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<br>state | Time left | Link Layer Address | Interface | vlan | prlvl | age  |
|--------------------------------|-----------|--------------------|-----------|------|-------|------|
| 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, 13-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, 13-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 Learned 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:

|   |   |       |             |             |      |     |                     |
|---|---|-------|-------------|-------------|------|-----|---------------------|
| → | 3 | 0.000 | 10.12.12.12 | 10.13.13.13 | ICMP | 164 | Echo (ping) request |
| ← | 4 | 0.000 | 10.13.13.13 | 10.12.12.12 | ICMP | 164 | Echo (ping) reply   |
|   | 5 | 0.000 | 10.12.12.12 | 10.13.13.13 | ICMP | 164 | Echo (ping) request |
|   | 6 | 0.000 | 10.13.13.13 | 10.12.12.12 | ICMP | 164 | Echo (ping) reply   |

```

▶ Frame 3: 164 bytes on wire (1312 bits), 164 bytes captured (1312 bits) on interface 0
▶ Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
▶ Internet Protocol Version 4, Src: 10.255.1.1, Dst: 10.255.2.1 ← Tunnel Endpoint IPs
▶ User Datagram Protocol, Src Port: 65478 (65478), Dst Port: 4789 (4789)
▼ Virtual eXtensible Local Area Network
  ▶ Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 99999 ← L3 VNI 99999 VRF vxlan
    Reserved: 0
  ▶ Ethernet II, Src: 00:01:00:01:00:00, Dst: 74:86:0b:c4:b7:5d ← VTEP-2 Dst: RMAC
  ▶ Internet Protocol Version 4, Src: 10.12.12.12, Dst: 10.13.13.13
  ▶ Internet Control Message Protocol
    
```

# 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 4: 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? | <p>Check the local subnets (or the SVI interfaces) if the remote subnet is present.</p> <p><b>Note</b> Local subnet has the remote subnet listed even in the case of scenario 3.</p> |
| Is the EVPN route type 5 being used to send traffic to remote destination?                               | <p>Run the <b>show bgp l2vpn evpn all</b> command in privileged EXEC mode on the VTEP. Look for remote prefix to be displayed as [ 5 ] for route type 5.</p>                         |

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 19](#) 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 31](#)
2. [Check the Table Entries for BGP, EVPN, and VPNv4 Tables, on page 31](#)
3. [Confirm that Adjacency exists to Reach Tunnel Destination, on page 34](#)

### 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  ports  siHandle
   riHandle                               diHandle  *a_time *e_time
-----
99    7035.0956.7edd                         0x8002   0      0      64  0x7ffa48d61be8  0x7ffa48d630b8
      0x0                                  0x5154   0      0      0
99    7486.0bc4.b75d                         0x1000001 0      0      64  0x7ffa48fb1bb8  0x7ffa48fac698
      0x7ffa48fab038                       0x7ffa4838cc18 0      0      0 RLOC 10.255.2.1 adj_id
103
99    ec1d.8b55.f55d                         0x1000001 0      0      64  0x7ffa48d065e8  0x7ffa48d01d08
      0x7ffa48c9a618                       0x7ffa4838cc18 0      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
```