# VXLAN Static Routing

VXLAN static routing provides a method for connecting multiple servers in a data center to an enterprise edge router. This chapter covers these topics:



# VXLAN Static Routing

***Table 1: Feature History Table***

| Feature Name | Release Information | Feature Description |
|---|---|---|
| VXLAN Static Routing | Release 25.4.1 | Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*)<br>*This feature is supported on Cisco 8711-48Z-M routers. |
| VXLAN Static Routing | Release 25.1.1 | Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*)<br>*This feature is supported on Cisco 8011-4G24Y4H-I routers. |

| Feature Name | Release Information | Feature Description |
|---|---|---|
| VXLAN Static Routing | Release 24.4.1 | Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, P100])(select variants only*)<br>The VXLAN Static Routing functionality is now extended to:<br>• 8712-MOD-M<br>• 8212-48FH-M<br>• 8711-32FH-M<br>• 88-LC1-52Y8H-EM<br>• 88-LC1-12TH24FH-E<br>• 88-LC1-36EH |

| Feature Name | Release Information | Feature Description |
|---|---|---|
| VXLAN Static Routing | Release 24.2.11 | You can now configure the source and destination virtual tunnel endpoints (VTEPs) for a particular traffic flow, which is particularly useful for scenarios where your data center is connected to an enterprise network, so multiple servers in the data center provide cloud services to your customers and the enterprise edge router. These endpoints help provide rapid convergence in case of failure. Plus, using the UDP header in the VXLAN packet, the VXLAN static routing (also called unicast VXLAN) facilitates network balancing by preventing the transmission of replicated packets.<br><br>Alternatively, you can use Service Layer API for faster provisioning of VXLAN static routing.<br><br>This feature is supported only on the following PIDs:<br>• 8202-32FH-M<br>• 8101-32H<br>• 8201-32FH<br><br>This feature introduces these changes:<br>• **CLI:**<br>• **host-reachability protocol static**<br>• **overlay-encapsulation**<br>• **hw-module profile cef vxlan ipv6-tnl-scale**<br>• **YANG Data Model:** (see GitHub, YANG Data Models Navigator)<br>• `Cisco-IOS-XR-tunnel-nve-cfg`<br>• `Cisco-IOS-XR-ip-static-cfg` |

## Introduction to VXLAN

Traditionally, Virtual Local Area Networks (VLANs) are used to partition a single physical network into multiple logical networks. With VLANs, every VLAN has a VLAN ID, which is added to a frame to keep traffic unique. The VLAN ID is 12-bits long, allowing around 4000 unique VLANs.

But in today's networks, you might have a data center with lots of virtualization and need to isolate several virtual machines (VMs) from other VMs where you could easily run out of VLANs. So, there is a need to provide robust tunneling mechanisms to isolate and load-balance traffic inside the provider’s network.

Virtual Extensible LAN (VXLAN) addresses some of the limitations of traditional VLANs in large-scale and cloud-based environments. VXLAN is widely used in data center environments where there is a need for virtualized networks to support cloud computing and virtualization technologies. It is also used in service provider networks to provide virtualized network services to customers.

VXLAN is a tunneling protocol that stretches Layer 2 networks over an underlying Layer 3 IP network. The VXLAN tunnel endpoint (VTEP) encapsulates and de-encapsulates Layer 2 traffic. VTEP encapsulates Layer 2 Ethernet frames within the Layer 4 User Datagram Protocol (UDP) and transports the encapsulated frames over a Layer 3 network.

VXLAN introduces an 8-byte VXLAN header that consists of a 24-bit VXLAN network identifier (VNI) with the original Ethernet frame added in the UDP payload. The 24-bit VNI is used to identify Layer 2 segments and maintain Layer 2 isolation between the segments.

With all 24 bits in VNI, VXLAN can support 16 million LAN segments. The VNI is used to designate the individual VXLAN overlay network on which the communicating virtual machines (VMs) are situated. VMs in different VXLAN overlay networks cannot communicate with each other.

VXLAN is a Layer 2 tunneling protocol that connects multiple servers in a data center that provide cloud services to customers and the enterprise edge router. VXLAN automatically configures underlay tunnels between the router and servers and overlay routing within those tunnels. VXLAN creates virtual networks on top of an underlay network. The underlay network is typically a physical IP network. VXLAN underlay can be IPv4 or IPv6 packets. The underlay and overlay networks are independent, and changes in the underlay don't affect the overlay. You can add or remove a router in the underlay network without affecting the overlay network.

VXLAN allows you to tunnel Ethernet frames over IP transport that uses IP and UDP as the transport protocol. A tunnel is created that enables you to extend a Layer 2 segment over a Layer 3 network using MAC-in-UDP encapsulation. A VXLAN header is added to the Layer 2 frame and placed inside a UDP packet to send to the routed domain. The VXLAN adds Layer 2 header, and the remote endpoint can ignore this header. The VXLAN tunnel endpoint (VTEP) is a router that encapsulates and de-encapsulates Layer 2 traffic.

When a host sends traffic:

- The VXLAN encapsulates the traffic in UDP and IP headers.
- VXLAN encodes the flow information in the UDP source port to enable routers to perform flow-based load balancing.

  Flow-based load balancing identifies different flows of traffic based on the key fields in the data packet. For example, IPv4 source and destination IP addresses can be used to identify a flow.
- VXLAN encapsulates these packets into the tunnel with an IPv4 or IPv6 outer header.
- After the traffic reaches the destination router, the router decapsulates the packet and sends it to the destination host.
- VXLAN adds the custom source MAC address in the inner header that encodes the information in the MAC address where your internal network devices can extract the required information.

For more information on VXLAN, see Key Concepts, on page 7.

### Benefits of VXLAN

VXLAN provides the following benefits:

- High throughput through dedicated VPN connectivity between servers and enterprise edge routers.
- Allows the creation of overlay networks independent of the underlying physical network, which provides greater flexibility in network design and deployment.
- Flexible placement of multitenant segments throughout the data center with the creation of isolated virtual networks for multiple tenants, providing greater security and separation between different users.

- Extends Layer 2 segments over the underlying shared network infrastructure to manage tenant workloads across physical pods in the data center.
- Uses a 24-bit VXLAN Network Identifier (VNI) which enables the creation of up to 16 million unique virtual networks, providing greater scalability.
- Facilitates network load balancing using the source UDP port within the VXLAN outer header.

### VXLAN Static Routing

You can use VXLAN static routing to interconnect non-VXLAN, such as MPLS and VXLAN domains. VXLAN static routing defines the path for VXLAN traffic from the source VTEP to reach the destination VTEP and involves configuring static routes on the underlying Layer 3 network to direct the VXLAN traffic to the appropriate VTEPs.

VXLAN supports up to 160000 static routes by default. However, you can increase the scale value up to one million VXLAN static routes for IPv6 tunnel remote nexthop using the **hw-module profile cef vxlan ipv6-tnl-scale** command.

### Benfits of VXLAN Static Routing

- You can use static routes in scenarios where consistent routing decisions are required. Because the static routes are manually configured and the routing behavior is predictable and stable.
- You can specify the next hop for each destination using static routes and thereby have direct control over traffic.
- Static routes are useful for specific traffic engineering or policy requirements.
- You do not have to maintain routing tables for static routing, hence reduces any overhead associated with routing protocols.

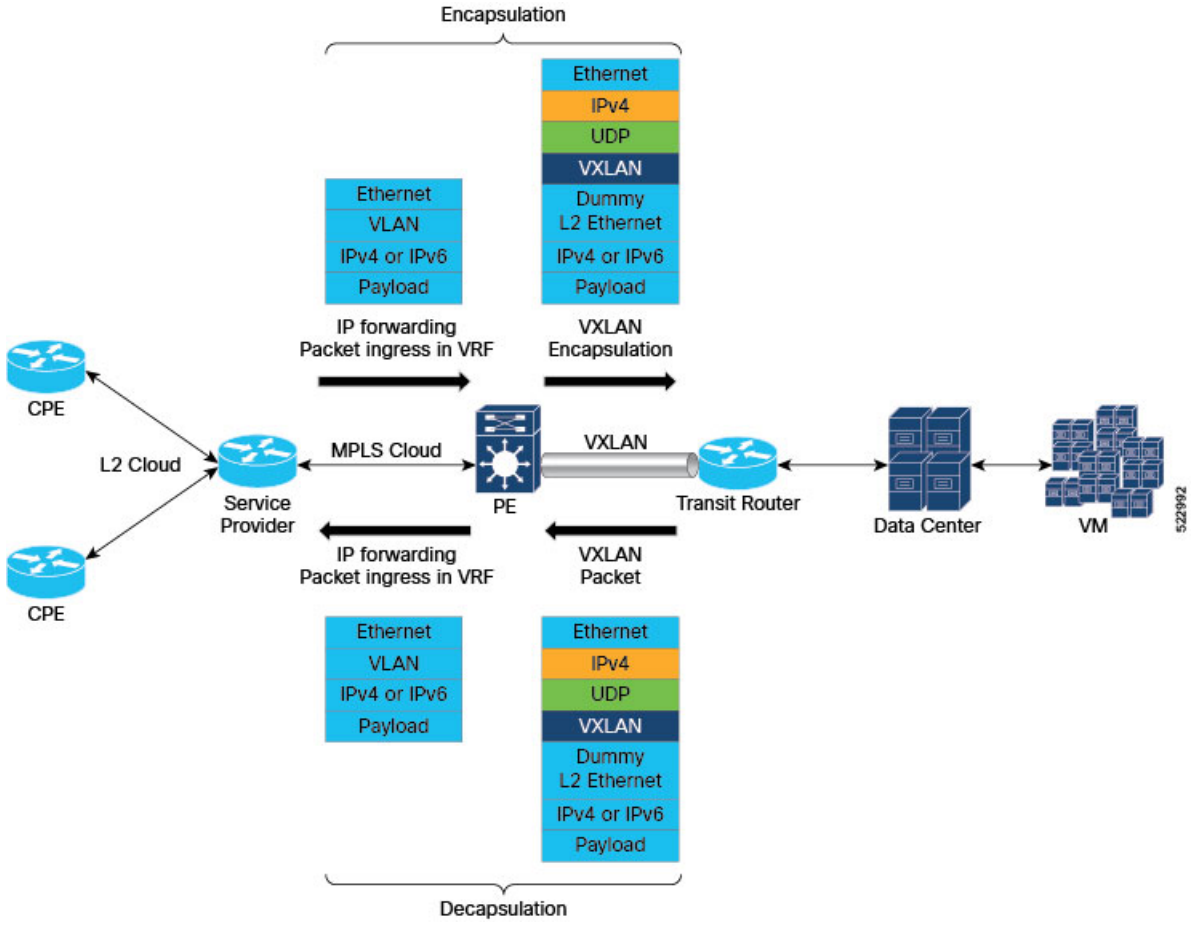
### Restrictions for VXLAN Static Routing

- Identical remote next hops with the same NVE with different VNI is not supported.
- Two same remote next hops in the same VRF, when the VNI is different, are not supported. However, same next hops with different NVE is supported.
- The index can be the same within a single VRF for the same VNI. However, the same VNI for different VRFs, must have a different index.
- Source MAC address in the static encapsulation route cannot be a multicast address.

### Topology

Let’s understand how VXLAN static routing works using this topology.

VXLAN Static Routing




In this topology,

- The PE router receives Layer 3 traffic on the VRF interface.
- The VXLAN tunnel starts at the PE router and terminates on the transit router or the servers behind the transit router.
- A BGP session is established over the tunnel.
- When Layer 3 traffic enters the PE router from the CPE, the PE router encapsulates the packet into the VXLAN tunnel and sends the traffic to the transit routers. Traffic that enters the PE has the VLAN tags for tracking customers within your network domain. VLANs map to VRF on the PE and to the VXLAN VNI. VXLAN encapsulation can be IPv4 or IPv6.
- The traffic is distributed based on your configured UDP source port. UDP source port value ranges from 49152 to 65535.
- The VXLAN tunnel terminates at the transit router.
- The transit router decapsulates the packet and performs an IP lookup to route traffic to the customer VM.

Similarly, the traffic from the VM is encapsulated as VXLAN. There will be an additional L2 header in the packet that must be terminated. Both VXLAN and the inner L2 header are terminated on the PE.

### VXLAN Static Routing using Service Layer API

You can configure VXLAN static routing using Service Layer API, which results in faster provisioning, easier scaling, and improved overall management of VXLAN networks.

Large cloud providers often need to provision tunneling mechanisms quickly and at a large scale to isolate end customer traffic dynamically. However, the traditional method of configuring the router through CLI might not be efficient. We provide Service Layer API as an alternate way to manage routers to address this challenge. With Service Layer API, you can have granular control of network traffic over the forwarding plane. It leverages Google's gRPC to generate client and server bindings, which allows users to program the forwarding plane using a variety of programming languages.

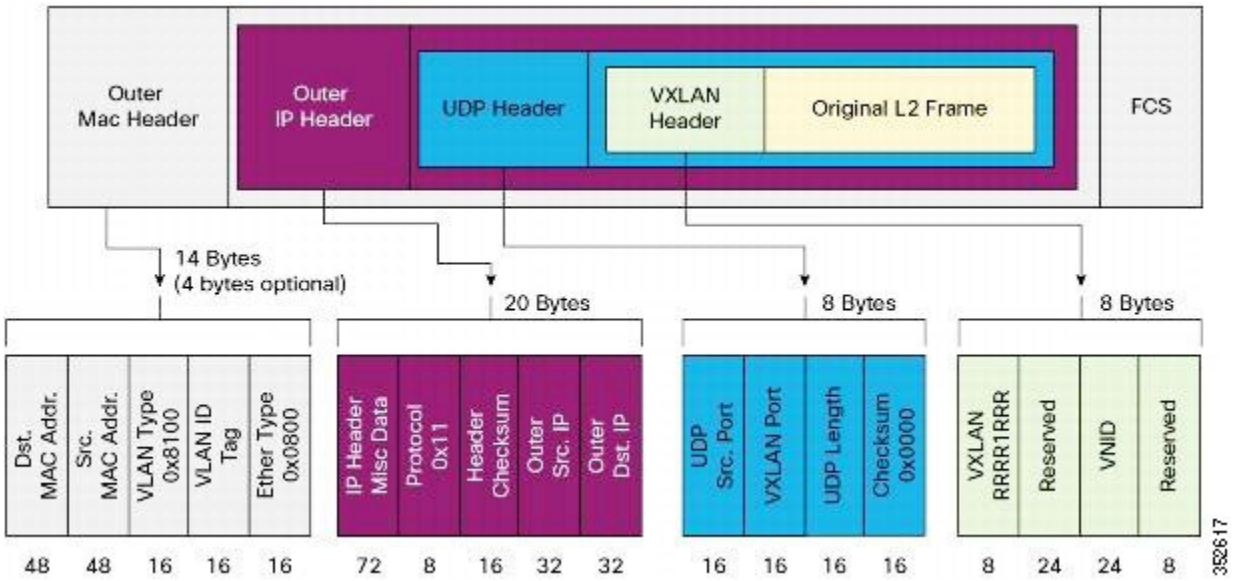
For more information on Service Layer API, see the *Use Service Layer API to Bring your Controller on Cisco IOS XR Router* chapter in the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

# Key Concepts

### VXLAN Packet Format

Here is the VXLAN packet format.

***Figure 1: VXLAN Packet Format***




### VXLAN Tunnel Endpoint

A VXLAN tunnel endpoint (VTEP) can be a physical or virtual router that connects the overlay and the underlay networks. A VTEP device is identified in the IP transport network using a unique IP address, which is a loopback interface IP address. The VTEP device uses this IP address to encapsulate Ethernet frames and transmits the encapsulated packets to the transport network through the IP interface. A source and destination VTEP creates a stateless tunnel to deliver traffic from one host to another. When a frame for a remote host reaches a device, the frame is encapsulated in IP and UDP headers. A maximum of 8k VXLAN tunnel interface per VTEP is supported.

### Load Sharing with VXLANs

Most data center transport networks are designed and deployed with multiple redundant paths that utilize various multipath load-sharing technologies to distribute traffic loads on all available paths. Encapsulated VXLAN packets are forwarded between VTEPs based on the native forwarding decisions of the transport network.

A typical VXLAN transport network is an IP-routing network that uses the standard IP equal cost multipath (ECMP) to balance the traffic load among multiple best paths. To avoid out-of-sequence packet forwarding, flow-based ECMP is commonly deployed. An ECMP flow is defined by the source and destination IP addresses.

All the VXLAN packet flows between a pair of VTEPs have the same outer source and destination IP addresses. All VTEP devices must use one identical destination UDP port, either the Internet Allocated Numbers Authority (IANA)-allocated UDP port 4789 or a customer-configured port. The source UDP port is the only variable element in the ECMP flow definition that can differentiate VXLAN flows from the transport network standpoint. The VXLAN outer-packet header uses source UDP port for link load-share hashing, which is the only element that can uniquely identify a VXLAN flow. A VXLAN flow is unique as the VXLAN inner frame header considers the VXLAN source UDP port for load balancing.

# Configure VXLAN Static Routing

Perform the following tasks on the provider edge (PE) router to configure VXLAN static routing:

- Configure VRF
- Configure interface NVE for decapsulation
- Configure static routing
- Configure customized UDP destination port and UDP source port range

```
/* Configure VRF */
Router# configure
Router(config)# vrf vrf1
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# import route-target
Router(config-vrf-import-rt)# 1:1
Router(config-vrf-import-rt)# exit
Router(config-vrf-af)# export route-target
Router(config-vrf-export-rt)# 1:1
Router(config-vrf-export-rt)# exit
Router(config-vrf-af)# exit
Router(config-vrf)# address-family ipv6 unicast
Router(config-vrf-af)# import route-target
Router(config-vrf-import-rt)# 1:1
Router(config-vrf-import-rt)# exit
Router(config-vrf-af)# export route-target
Router(config-vrf-export-rt)# 1:1
Router(config-vrf-export-rt)# root
Router(config)# vrf vrf2
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# import route-target
Router(config-vrf-import-rt)# 1:2
Router(config-vrf-import-rt)# exit
Router(config-vrf-af)# export route-target
Router(config-vrf-export-rt)# 1:2
Router(config-vrf-export-rt)# exit
```

```
Router(config-vrf-af)# exit
Router(config-vrf)# address-family ipv6 unicast
Router(config-vrf-af)# import route-target
Router(config-vrf-import-rt)# 1:2
Router(config-vrf-import-rt)# exit
Router(config-vrf-af)# export route-target
Router(config-vrf-export-rt)# 1:2
Router(config-vrf-export-rt)# commit

/* Configure interface NVE for decapsulation */
Router(config)# interface nve1
Router(config-if)# member vni 2
Router(config-nve-vni)# vrf vrf1
Router(config-nve-vni)# host-reachability protocol protocol static
Router(config-nve-vni)# exit
Router(config-if)# member vni 6
Router(config-nve-vni)# vrf vrf2
Router(config-nve-vni)# host-reachability protocol protocol static
Router(config-nve-vni)# exit
Router(config-if)# overlay-encapsulation vxlan
outer(config-nve-encap-vxlan)# peer-ip lookup disable
Router(config-nve-encap-vxlan)# exit
Router(config-if)# source-interface Loopback1
Router(config-if)# commit

/* Configure static routing */
Router# configure
Router(config)# router static
Router(config-static)# address-family ipv4 unicast
Router(config-static-afi)# 10.10.10.10/32 10.151.11.2
Router(config-static-afi)# 10.10.10.11/32 10.151.11.2
Router(config-static-afi)# exit
Router(config-static)# vrf VRF1
Router(config-static-vrf)# address-family ipv4 unicast
Router(config-static-vrf-afi)# 11.1.1.1/32 remote-next-hop 10.10.10.10 tunnel VXLAN index
1 nve 1 evni 1 src-mac aaa1.bbb1.ccc1 ->  IPv4 over IPv4
Router(config-static-vrf-afi)# 11.1.1.2/32 remote-next-hop 10:10:10::10 tunnel VXLAN index
 3 nve 1 evni 3 src-mac aaa1.bbb1.ccc1 -> IPv4 over IPv6
Router(config-static-vrf-afi)# exit
Router(config-static-vrf)# exit
Router(config-static)# address-family ipv6 unicast
Router(config-static-afi)# 10:10:10::10/128 10:151:11::2
Router(config-static-afi)# 10:10:10::11/128 10:151:11::2
Router(config-static-afi)# exit
Router(config-static)# vrf VRF1
Router(config-static-vrf)# address-family ipv6 unicast
Router(config-static-vrf-afi)# 11:1:1::1/128 remote-next-hop 10.10.10.11 tunnel VXLAN index
 2 nve 1 evni 2 src-mac aaa1.bbb1.ccc1 ->  IPv6 over IPv4
Router(config-static-vrf-afi)# 11:1:1::2/128 remote-next-hop 10:10:10::11 tunnel VXLAN index
 4 nve 2 evni 4 src-mac aaa1.bbb1.ccc1 -> IPv6 over IPv6
Router(config-static-vrf-afi)# commit

/* Configure customized UDP destination port and UDP source port range */
Router# configure
Router(config)# nve
Router(config-nve)# overlay-encap vxlan
Router(config-vxlan)# udp-port destination 65330
Router(config-vxlan)# udp-port src-port start 1024
Router(config-vxlan)# commit
```

### Running Configuration

This section shows VXLAN static routing running configuration.

```
/* Configure VRF */
vrf vrf1
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
 !
 address-family ipv6 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
vrf vrf2
 address-family ipv4 unicast
  import route-target
   1:2
  !
  export route-target
   1:2
  !
 !
 address-family ipv6 unicast
  import route-target
   1:2
  !
  export route-target
   1:2

/* NVE and  Encapsulation*/
interface nve1
 member vni 2
  vrf vrf1
  host-reachability protocol protocol static
 !

 member vni 6
  vrf vrf2
  host-reachability protocol protocol static
 !
 overlay-encapsulation vxlan
 peer-ip lookup disable
!
 source-interface Loopback0


/* Static Routing */
router static
address-family ipv4 unicast
  10.10.10.10/32 10.151.11.2
  10.10.10.11/32 10.151.11.2
!
address-family ipv6 unicast
  10:10:10::10/128 10:151:11::2
  10:10:10::11/128 10:151:11::2
```

```
vrf vrf1
  address-family ipv4 unicast
   11.1.1.1/32 remote-next-hop 10.10.10.10 tunnel VXLAN index 1 nve 1 evni 1 src-mac
aaa1.bbb1.ccc1. ====> IPv4oIPv4

   11.1.1.2/32 remote-next-hop 10:10:10::10 tunnel VXLAN index 3 nve 1 evni 3 src-mac
aaa1.bbb1.ccc1 ====> IPv4oIPv6
  !
  address-family ipv6 unicast
   11:1:1::1/128 remote-next-hop 10.10.10.11 tunnel VXLAN index 2 nve 1 evni 2 src-mac
aaa1.bbb1.ccc1 ====> ipv6 over ipv4

   11:1:1::2/128 remote-next-hop 10:10:10::11 tunnel VXLAN index 4 nve 2 evni 4 src-mac
aaa1.bbb1.ccc1 ====> ipv6 over ipv6


/* Decapsulation */
configure
 nve
  overlay-encap vxlan
   udp-port destination 65330
   udp-port src-port start 1024
  !
 !
```

### Verification

Verify that the NVE interface is Up.

```
Router# show nve interface nve 1
Interface: nve1 State: Up Encapsulation: VxLAN
  Source Interface: Loopback0 (primary: v4: 1.1.1.1 v6: 1:1:1::1)

Router# show nve global

 NVE Global details
   VNI Scope Local : No
   VxLAN Src Port  : 1024
   VxLAN Destination Port : 65330
   VxLAN interfaceless l3vni bring up: TRUE
   Count of NVE interfaces with mpls-udp encap: 0
   Global system mac: 9c54.1643.f900
```

Verify the static routing configuration.

```
Router# show route vrf VRF1 11.1.1.1 detail

Routing entry for 11.1.1.1/32
  Known via "static", distance 1, metric 0
  Installed Nov 30 17:44:37.003 for 02:22:09
  Routing Descriptor Blocks
    10.10.10.10, via Bundle-Ether13.5
      Nexthop in Vrf: "default", Table: "default", IPv4 Unicast, Table Id: 0xe0000000
      Route metric is 0
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      NHID:0x0(Ref:0)
```

```
      IP Tunnel Info: Auto create: 1 Tunnel Type: vxlan-l3 eVNI: 0x1 Tunnel ID: 0x1 RTEP
ID: 0x100000000000001
      MPLS eid:0xfffffffffffffffd
  Route version is 0x1 (1)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_RECURSIVE (9) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 3, Download Version 12
  Route eid: 0xfffffffffffffffd
  No advertising protos.
```

## VXLAN Static Routing using Service Layer API

### Configuration Example

```
Router# configure
Router(config)# grpc
Router(config-grpc)# port 57777
Router(config-grpc)# address-family ipv4
Router(config-grpc)# service-layer
Router(config-grpc)# no-tls
Router(config-grpc)# commit
```

### Running Configuration

```
grpc
 port 57777
 address-family ipv4
 service-layer
 no-tls
!
```

### Verification

Verify the service-layer state.

```
Router# show service-layer state

------------service layer state----------
config on:                    NO
connected to RIB for IPv4:    YES
connected to RIB for IPv6:    YES
Initialization state:         initialized
pending requests:             0
BFD Connection:               DOWN
MPLS Connection:              DOWN
Interface Connection:         UP
Objects accepted:             NO
interface registered:         NO
bfd registered for IPv4:      NO
bfd registered for IPv6:      NO
```