

# VxLAN Static Routing

VxLAN static routing provides a method for connecting multiple servers in a data center to an enterprise edge router. The method:

- Creates one-to-many static routes between the servers and enterprise edge router
- Automatically generates a point-to-multipoint (P2MP) VxLAN tunnels on the static routes on demand


- Feature Information for VxLAN Static Routing, on page 1
- Prerequisites for VxLAN Static Routing, on page 3
- Notes and Limitations for VxLAN Static Routing, on page 3
- Information About VxLAN Static Routing, on page 3
- How to Configure VxLAN Static Routing, on page 5
- How to Configure VxLAN Policing and Accounting, on page 10


# Feature Information for VxLAN Static Routing

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

*Table 1: Feature Information for VxLAN Static Routing*

| Feature Name | Releases | Feature Information |
|---|---|---|
| VxLAN Static Routing | Cisco IOS XE Gibraltar 16.10.1 | VxLAN static routing configures P2MP underlay tunnels between the router and multiple servers, and overlay routing within those tunnels. VxLAN static routing supports GPE and dummy-L2 P2MP tunnels.<br>The prefix supports only IPv4. The next hop supports IPv4 and IPv6.<br>The following commands were modified or added by this feature: **vxlan route-profile** , **show vxlan route-profile all**, **show vxlan static-route**, **show vxlan static-route next-hop bind-label**. |
| VxLAN Static Routing | Cisco IOS XE Gibraltar 16.11.1 | Added point-to-many point “ingress” tunnels from multiple servers to the router.<br>Added ability to specify a range of numbers for naming the tunnels, such as beginning with Tunnel100 and ending with Tunnel1000. The number is incremented by 1 for each new tunnel.<br>Added **show** commands that provide packet/byte statistics. |
| IPv6 Prefix for VxLAN Static Route | Cisco IOS XE Gibraltar 16.12.x | The VxLAN tunnels that operate at more than 10 gbps provide different types of encapsulations- IPv4-over-IPv4, IPv4-over-IPv6, IPv6-over-IPv4, and IPv6-over-IPv6. |
| VNET/VNI VRF Policing and Accounting | Cisco IOS XE Amsterdam 17.1 | Added information on VNET/VNI per-vrf policing and accounting.<br>Added the following commands:<br>• vxlan static-route policy output<br>• vrf vrf1<br>• police match any vni 1 rate |

| Feature Name | Releases | Feature Information |
|---|---|---|
| **Ingress and Egress Accounting MIB** | Cisco IOS XE Amsterdam 17.2 | newly added per VNI/VNET per VRF ingress and engress accounting MIB allows the user to access the counts by SNMP |

# Prerequisites for VxLAN Static Routing

- Underlay protocol, such as OSPF or IS-IS

# Notes and Limitations for VxLAN Static Routing

- Tunnels initiating using this method can encapsulate, but not decapsulate packets. The tunnel carries packets from the customer side to the cloud service side. A static tunnel carry packets sent from the cloud service to the customer side.
- You cannot modify a route profile if it is in use. It is considered to be in use if a tunnel that was created using the method described here is currently open.

# Information About VxLAN Static Routing

## Overview of VxLAN Static Routing

VxLAN static routing configures P2MP underlay tunnels between the router and multiple servers, and overlay routing within those tunnels. This connects multiple servers in a data center to the enterprise edge router. VxLAN static routing supports GPE and dummy-L2 P2MP tunnels.

VxLAN static routing provides a method for connecting multiple servers in a data center to an enterprise edge router. VxLAN static routing supports GPE and dummy-L2 P2MP tunnels.

This method:

- Creates one-to-many static routes between the servers and enterprise edge router
- Automatically generates VxLAN tunnels on the static routes on demand

A use case is, connecting the servers that provide cloud services to customers and the enterprise edge routers, such as a Cisco ASR 1000 Series router, that communicates with customers.

Advantages:

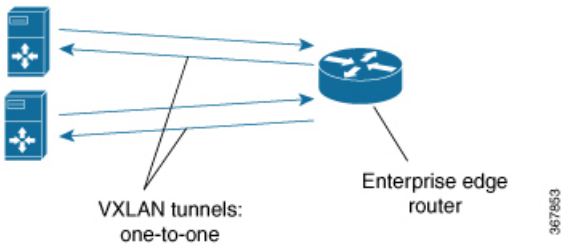
- High throughput dedicated VPN connectivity between servers and enterprise edge router
- Low latency
- Predictable performance (helps to meet service level agreements)
- High availability

- 

### Method

An earlier method of connecting multiple servers to a single enterprise edge router was numerous P2P connections.

***Figure 1: One-to-One VxLAN Tunnels***




VxLAN P2MP Tunnels provide one-to-many VxLAN tunnels in the router-to-server direction (egress), or in both the router-to-server and server-to-router (ingress) directions. The VxLAN tunnels operate at more than 10 Gbps, and can provide different types of encapsulation including IPv4-over-IPv4, IPv4-over-IPv6, IPv6-over-IPv4, and IPv6-over-IPv6.

***Figure 2: One-to-Many VxLAN Tunnels, Egress Only***

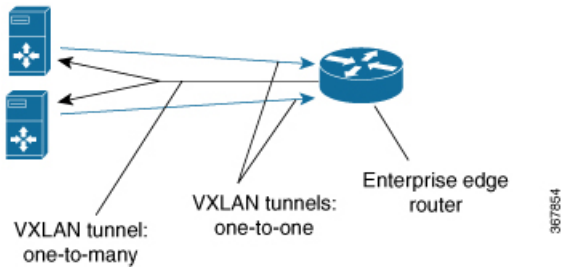
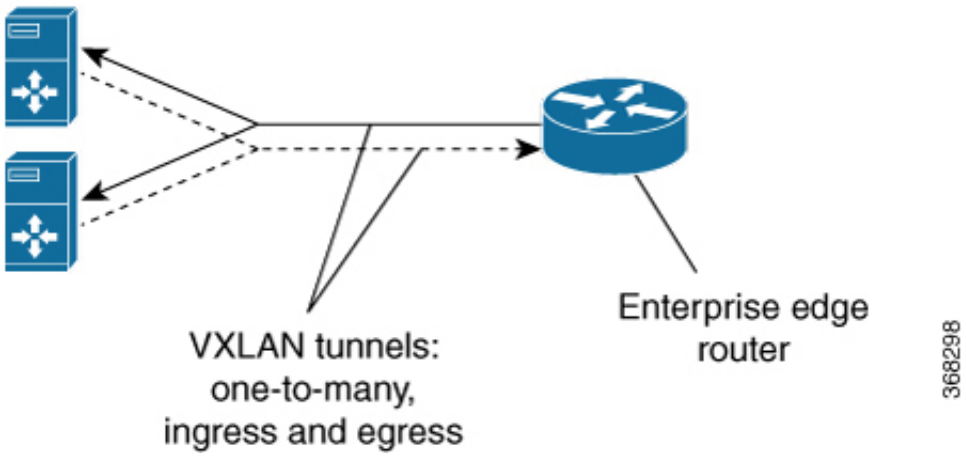



***Figure 3: One-to-Many VxLAN Tunnels, Egress and Ingress***




Typically, a network controller is used to manage the enterprise edge router and initiate the tunnel connections. The overall architecture for the cloud services use case is as follows:

*Figure 4: Complete Architecture, Egress Only Configuration*

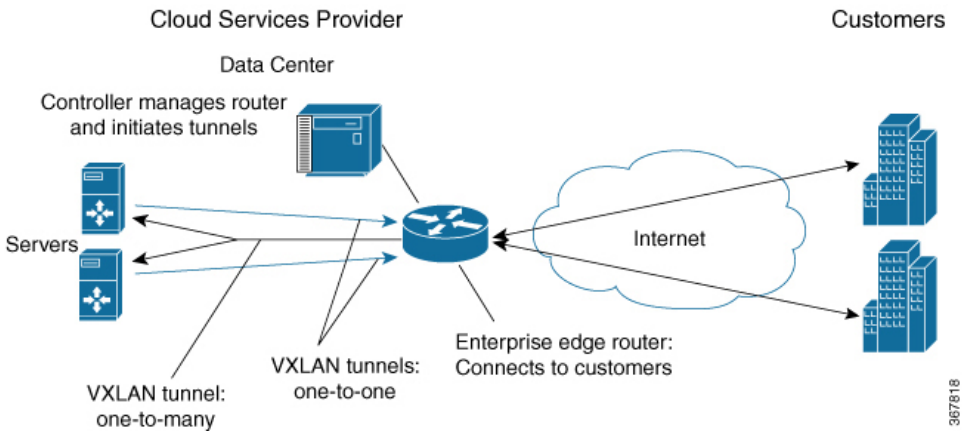
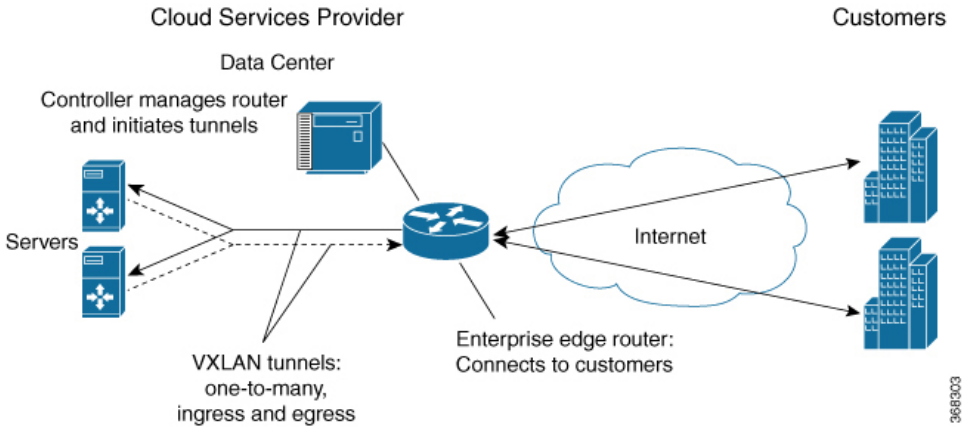



*Figure 5: Complete Architecture, Egress and Ingress Configuration*




# How to Configure VxLAN Static Routing

## Configuring VxLAN Static Routing

Perform this procedure on a router. A remote controller can initiate the tunnels.

1. Use **vxlan route-profile** to define a profile to use when creating new tunnels.
2. Use **vxlan static-route** to define multiple endpoints (servers).

### Before you begin

**Prerequisites**: Underlay protocol, such as OSPF or IS-IS

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **vxlan route-profile** *route-profile-name* **tunnel source interface** *interface* [**default-mac-source** *mac-address*] [**dscp** *dscp*] [**dst-port** *port*] [**tunnel mode** *mode*] [**tunnel mtu** *mtu*] [**tunnel source-port-hash** *hash*] [**tunnel source-port range** *port-range*] [**vxlan-reserved-word-1** *0x0000*] [**vxlan-reserved-word-2** *0x0000*] [**persistent**]<br>**Example:**<br>`vxlan route-profile af11`<br>`dscp 10`<br>`vxlan-reserved-word-1 1111`<br>`vxlan-reserved-word-2 17`<br>`tunnel mode vxlan-default-mac`<br>`tunnel source interface Loopback0`<br>`default-mac-source 0011.0011.0011`<br>`persistent` | Creates a route-profile.<br>For tunnel mode, there are two options:<br>• vxlan-gpe<br>• vxlan-default-mac<br>**Note**<br>A route profile cannot be modified if it is in use. It is considered to be in use if a tunnel that was created using the method described here is currently open. |
| **Step 2** | **vxlan static-route**<br>**Example:**<br>`vxlan static-route` | Creates a static route for one or more end-points (servers). In the following step, add a separate vrf line for each end-point. |
| **Step 3** | **vxlan static-route auto-tunnel range** *start end*<br>**Example:**<br>`vxlan static-route auto-tunnel range 100 100000` | (Optional) |
| **Step 4** | **vrf** *vrf-name IPv4/IPv6-address* {*mask-format-X.X.X.X* \| *mask-format-/XX*} **vni** *1-16777215 next-hop-IPv4/IPv6* **dst-mac** *MAC-Address* **route-profile** *route-profile-name* [**symmetric**] [ **persistent**]<br>**Example:**<br>`vxlan static-route`<br>`vrf host1_1_1 100.0.10.0 255.255.255.0 vni 1 11.11.11.11 dst-mac 0011.0000.0010 route-profile af11 persistent`<br>`vrf host1_1_1 100.0.10.0 255.255.255.0 vni 1 1111::1111 dst-mac 0011.0000.0010 route-profile af11 persistent`<br>`vrf host1_1_1 100.0.20.0 255.255.255.0 vni 1 11.11.11.11 dst-mac 0011.0000.001a route-profile af21 persistent`<br>`vrf host1_1_1 100.0.20.0 255.255.255.0 vni 1 1111::1111 dst-mac 0011.0000.001a route-profile af21 persistent` | Use the *route-profile-name* defined in an earlier step.<br>persistent: Save the configuration line in the device NVRAM, persists even if device reboots.<br>symmetric: Use the configured point-to-many-point tunnel for ingress also. Applies only to the specified vni.<br>The prefix used for overlay supports only IPv4 addresses . Tthe next hop used for underlay supports IPv4 and IPv6. |

# Examples

### Example: Point-to-many-point Tunnel for Egress

The **vxlan route-profile** defines a profile to use when creating new tunnels.

The **vrf** lines following **vxlan static-route** define multiple endpoints (servers).

```
vxlan route-profile af11
dscp 10
vxlan-reserved-word-1 1111
vxlan-reserved-word-2 17
tunnel mode vxlan-default-mac
tunnel source interface Loopback0
default-mac-source 0011.0011.0011
persistent

vxlan static-route
vrf host1_1_1 100.0.10.0 255.255.255.0 vni 1 11.11.11.11 dst-mac 0011.0000.0010 route-profile
 af11 persistent
vrf host1_1_1 100.0.10.0 255.255.255.0 vni 1 1111::1111 dst-mac 0011.0000.0010 route-profile
 af11 persistent
vrf host1_1_1 100.0.20.0 255.255.255.0 vni 1 11.11.11.11 dst-mac 0011.0000.001a route-profile
 af21 persistent
vrf host1_1_1 100.0.20.0 255.255.255.0 vni 1 1111::1111 dst-mac 0011.0000.001a route-profile
 af21 persistent
```

### Example: Symmetric Point-to-many-point Tunnels for Egress and Ingress

The **vxlan route-profile** line defines a profile to use when creating new tunnels.

The **vxlan static-route auto-tunnel range** line sets the range for numbering of tunnel names.

The **vrf** lines following **vxlan static-route** define multiple endpoints (servers), creating symmetric point-to-many-point tunnels in both directions between the servers and the router.

```
vxlan route-profile af11
dscp 10
vxlan-reserved-word-1 1111
vxlan-reserved-word-2 17
tunnel mode vxlan-default-mac
tunnel source interface Loopback0
default-mac-source 0011.0011.0011
persistent

vxlan static-route auto-tunnel range 100 100000

vxlan static-route
vrf host1_1_1 100.0.10.0 255.255.255.0 vni 1 11.11.11.11 dst-mac 0011.0000.0010 route-profile
 af11 symmetric persistent
vrf host1_1_1 100.0.10.0 255.255.255.0 vni 1 1111::1111 dst-mac 0011.0000.0010 route-profile
 af11 symmetric persistent
vrf host1_1_1 100.0.20.0 255.255.255.0 vni 1 11.11.11.11 dst-mac 0011.0000.001a route-profile
 af21 symmetric persistent
vrf host1_1_1 100.0.20.0 255.255.255.0 vni 1 1111::1111 dst-mac 0011.0000.001a route-profile
 af21 symmetric persistent
```

# Viewing VxLAN Static Routing Status

## SUMMARY STEPS

1. **show vxlan route-profile all**
2. **show vxlan static-route** { all | summary | vrf *vrf-name* }
3. **show vxlan route-profile name** *profile-name* **auto-tunnel**
4. **show vxlan static-route next-hop bind-label** *tunnel-id*
5. **show vxlan static-route statistics vrf test all detail**
6. **show vxlan static-route statistics vni** *vni* **detail**

## DETAILED STEPS

### Procedure

**Step 1** **show vxlan route-profile all**

Displays all route-profile configurations.

**Step 2** **show vxlan static-route** { all | summary | vrf *vrf-name* }

Displays VxLAN static route configurations.

**Example:**

Example of summary output:

```
Device# show vxlan static-route summary
vxlan static-route summary:
prefix count: 6
persistent prefix count: 5
route-profile count: 2
vxlan next-hop count: 8
vxlan auto-tunnel count: 4
vxlan auto-tunnel range: [200000, 300000]
default dst mac: 0000.5e00.5214
```

**Example:**

Example of detailed output for a specific VRF:

```
Device# show vxlan static-route vrf test 2.2.2.8/32 detailed
vrf test2 2.2.2.8/32 vni 8 3.3.3.2 route-profile test2, binding_label: 0x2000008, connection_id: 8
vrf test2 2.2.2.8/32 vni 8 3.3.3.3 route-profile test2, binding_label: 0x2000006, connection_id: 6
vrf test2 2.2.2.8/32 vni 8 3.3.3.3 dst-mac 1212.1212.1212 route-profile test2, binding_label: 0x2000007,
 connection_id: 7
```

**Step 3** **show vxlan route-profile name** *profile-name* **auto-tunnel**

Displays any active tunnels that have been generated automatically using the specified route profile. Tunnel IDs are generated automatically, numbered consecutively within a preset range.

**Note**
If there are active tunnels using a route profile, the route profile cannot be altered.

**Example:**

```
Device# show vxlan route-profile name test auto-tunnel
Vxlan Route Profile test:
  IPv4 auto tunnel: Tunnel200000
  IPv6 auto tunnel: Tunnel200001
```

**Step 4** **show vxlan static-route next-hop bind-label** *tunnel-id*

Displays the details of the next-hop (server address) for a specific IP static route which is identified by a hexadecimal bind-label. Use **show ip route** to display the routes that have been configured, and the bind-labels for each route.

**Example:**

This example uses **show ip route** to display the routes on the route_symmetric VRF. It displays details for the route with a bind-label of 0x2000002 (in the output highlighted the binding label 0x2000002).

```
Device# show ip route vrf route_symmetric
Routing Table: scale_route_symmetric
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR
Gateway of last resort is not set
      111.0.0.0/32 is subnetted, 91355 subnets
S        111.0.33.198 [1/0] via binding label: 0x2000002

Device# show vxlan static-route next-hop bind-label 0x2000002
vxlan static route next hop:
vni: 2
address: 20.1.1.1
auto interface: Tunnel0
route profile: test
connection-id: 2
bind-label: 0x2000002
refer count: 1
```

**Step 5** **show vxlan static-route statistics vrf test all detail**

Displays the packets and bytes sent over each VRF. This info is useful for accounting purposes.

**Note**
Enter the **ip cef accounting per-prefix** command before using this **show** command.

**Example:**

```
Device#ip cef accounting per-prefix
Device#show vxlan static-route statistics vrf test all detail
Vrf                  Prefix               Tx-Pkts      Tx-Bytes
test              100.0.30.0/24        0            0
test              100.0.30.2/32        3317         4630532
test              100.0.30.3/32        3317         4630532
------------------------------------------------------------
                                       6634         9261064
```

**Step 6** **show vxlan static-route statistics vni** *vni* **detail**

Displays the packets and bytes for a specific VNI. This info is useful for accounting purposes.

**Example:**

```
Device#show vxlan static-route statistics vni 100 detail
Vni    Next-hop         Intf            Route-profile   Pkts    Bytes
100    11.11.11.11      Tunnel200002    p1              111     15096
100    33:33:33::33     Tunnel200003    p1               50      7800
-------------------------------------------------------------------------------
                                                        161     22896
```

# How to Configure VxLAN Policing and Accounting

## VXLAN Accounting and Policing

The VxLAN policing and accounting can be enabled or disabled according to requirement. If you want to account and police the traffic both through VxLAN static route and VxLAN p2p tunnel enable it using **vxlan static-route accounting-policying bind p2p-tunnel** command.

### Before you begin

**Prerequisites**: Ensure that you have configured VRF and VxLAN static route.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **vxlan static-route policy output** | Enters VxLAN route policy mode. |
| **Step 2** | **vrf vrf1**<br>**Example:**<br>`vxlan static-route` | Enters config-vxlan-route-policy-vrf mode. |
| **Step 3** | **police match any vni** *vxlan identifier***rate***traffic bits per second*<br>**Example:**<br>`police match any vni 1 rate 8000` | Specifies the minimum traffic rate for each VNI. |

## Viewing Accounting and Policing Status

### SUMMARY STEPS

1. **show vxlan static-route vni-stats vrf vrf1 vni 1**
2. **show vxlan static-route policy vrf vrf1 vni 1**

**DETAILED STEPS**

**Procedure**

**Step 1** **show vxlan static-route vni-stats vrf vrf1 vni 1**

Displays all configurations for a specific VRF.

```
Device# show vxlan static-route vni-stats vrf vrf1 vni 1
VRF: vrf2
Vni Tx-Pkts Tx-Bytes Rx-Pkts Rx-Bytes 1 5000 7140000 50 71400
```

**Step 2** **show vxlan static-route policy vrf vrf1 vni 1**

Displays VxLAN static route configurations.

```
Device# show vxlan static-route policy vrf vrf1 vni 1
vrf: vrf1
vni: 1
ref count: 1
obj id: 3
rate: 500000 bps
bc: 15625 bytes
confirmed: 500 pkts,
7140000 bytes; action: transmit exceeded: 0 pkts, 0 bytes; action: drop
```

# Information About Ingress and Egress Accounting MIB

From Cisco IOS XE Amsterdam Release 17.2, newly added per VNI/VNET per VRF ingress and engress accounting MIB allows the user to access the counts by SNMP.

*Table 2: Ingress and Egress Accounting MIB for per VNI/VNET per VRF*

| Name | Description |
|---|---|
| **cnvoVNetVrfStatsTable** | Table containing all statistics information for the Per VNI/VNET per VRF ingress and egress accounting. |
| **cnvoVNetVrfStatsVrfName** | VRF Key to identity count instance. It is not accessible in MIB. |
| **cnvoVNetVrfStatsVni** | VNI Key to identity count instance. It is not accessible in MIB. |
| **cnvoVNetVrfIngressPackets** | Ingress packets count Per VNI/VNET per VRF. It is not writable. |
| **cnvoVNetVrfIngressBytes** | Ingress traffic bytes count Per VNI/VNET per VRF. It is not writable. |
| **cnvoVNetVrfEgressPackets** | Engress packets count Per VNI/VNET per VRF. It is not writable. |

| Name | Description |
|---|---|
| **cnvoVNetVrfEgressBytes** | Engress traffic bytes count Per VNI/VNET per VRF. It is not writable. |

### Configuring the SNMP

The following examples show how to configure SNMP.

Add each VxLAN OID MIB view into the SNMP view:

```
snmp-server view <view-name> cnvoVNetVrfEgressBytes included
snmp-server view <view-name> cnvoVNetVrfEgressPackets included
snmp-server view <view-name> cnvoVNetVrfIngressBytes included
snmp-server view <view-name> cnvoVNetVrfIngressPackets included
```

Or add the table into the SNMP view:

```
snmp-server view <view-name> cnvoVNetVrfStatsTable included
```

### Example: SNMP Request for ODI

These examples show the SNMP request for ODI:

**Request all ingress packets counters under vrf vrf1:**

```
snmpwalk -v 3 -u test -A testpassword -l authNoPriv -a md5 10.75.28.170
1.3.6.1.4.1.9.9.820.1.1.6.1.3.4.118.114.102.49
3:IngressPackets
4: vrf name length
118.114.102.49: vrf name
```

**Request ingress packets counters under vrf vrf1 vni 1:**

```
snmpget -v 3 -u test -A testpassword -l authNoPriv -a md5 10.75.28.170
1.3.6.1.4.1.9.9.820.1.1.6.1.3.4.118.114.102.49.1
1: vni id
```

**Request all counters:**

```
snmpwalk -v 3 -u test -A testpassword -l authNoPriv -a md5    10.75.28.170
1.3.6.1.4.1.9.9.820.1.1.6
```

**Request all ingress packets counters:**

```
snmpwalk -v 3 -u test -A testpassword -l authNoPriv -a md5 10.75.28.170
1.3.6.1.4.1.9.9.820.1.1.6.3
```

For more information on this MIB, see the SNMP Object Navigator.