VxLAN GPE P2MP Tunnels

VxLAN GPE (generic protocol extension) P2MP (point-to-multipoint) tunnels provide 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 VxLAN tunnels on the static routes on demand

Feature Information for VxLAN GPE P2MP Tunnels

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 GPE P2MP Tunnels

Feature Name	Releases	Feature Information
VxLAN GPE P2MP Tunnels	Cisco IOS XE Gibraltar 16.10.1	VxLAN GPE P2MP tunnels provide a method for connecting multiple servers in a data center to an enterprise edge router. 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**.

Prerequisites for VxLAN GPE P2MP Tunnels

- Underlay protocol, such as OSPF or ISIS

Notes and Limitations for VxLAN GPE P2MP Tunnels

- Tunnels initiating using this method can encapsulate, but not decapsulate packets. The tunnel carries packets from the customer side to the cloud service side. Packets sent from the cloud service to the customer side are carried by a static tunnel.
- 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.

Information About VxLAN GPE P2MP Tunnels

Overview of VxLAN GPE P2MP Tunnels

VxLAN GPE P2MP Tunnels provide a method for connecting multiple servers in a data center to an enterprise edge router. 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 ASR1000 Series router, that communicate with customers.

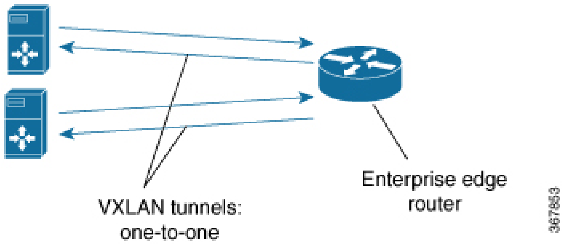
Advantages:

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

Method

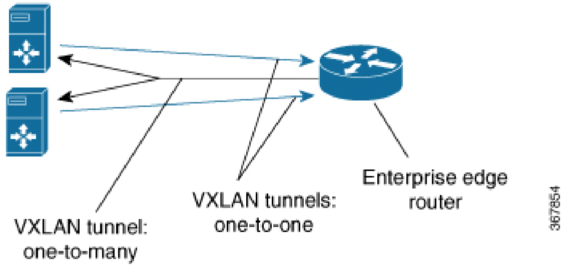
An earlier method of connecting multiple servers to a single enterprise edge router was numerous P2P connections.

Figure 1: One-to-One VxLAN Tunnels



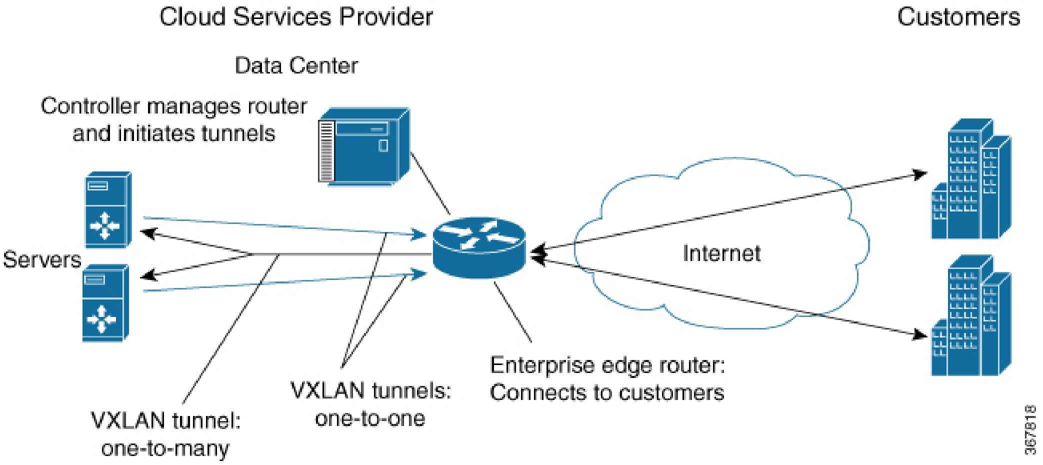
VxLAN GPE P2MP Tunnels provide one-to-many VxLAN tunnels in the router-to-server direction. The VxLAN tunnels operate at more than 10 Gbps, and can provide different types of encapsulation, including IPv4-over-IPv4 and IPv4-over-IPv6.

Figure 2: One-to-Many VxLAN Tunnels



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 3: Complete Architecture



How to Configure VxLAN GPE P2MP Tunnels

Configuring VxLAN GPE P2MP Tunnels

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 end-points (servers).

Before you begin

Prerequisites: Underlay protocol, such as OSPF or ISIS

SUMMARY STEPS

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**]
2. **vxlan static-route**
3. **vrf** *vrf-name IPv4-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* [**persistent**]

DETAILED STEPS

	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**]
	Example:
	`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`
		Creates a route-profile.
		Note 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** **Example:** `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.

	Command or Action	Purpose

Step 3	**vrf** *vrf-name IPv4-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* [**persistent**]
	Example:
	`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`
		Use the *route-profile-name* defined in an earlier step.

Example

The **vxlan route-profile** defines a profile to use when creating new tunnels. The **vrf** lines following **vxlan static-route** define multiple end-points (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
```

Viewing VxLAN GPE P2MP Tunnels 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*

DETAILED STEPS

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

Displays all route-profile configurations.

Example:

In the following example, the command shows the configuration of two profiles: profile1 and profile2 (bold added to show where each begins).

```
Device# show vxlan route-profile all
Vxlan route profile
name: profile1
dscp: 0 (default)
vxlan-reserved-word-1: 0x0 (default)
vxlan-reserved-word-2: 0x0 (default)
tunnel source-port-range: [49152, 65535] (default)
tunnel source-port-hash: 5-tuple (default)
tunnel mode: vxlan-gpe (default)
tunnel mtu: 1450 (default)
tunnel source interface: Loopback0
dst-port: 4790 (default)
persistent: yes
Vxlan route profile
name: profile2
dscp: 0 (default)
vxlan-reserved-word-1: 0x0 (default)
vxlan-reserved-word-2: 0x0 (default)
tunnel source-port-range: [49152, 65535] (default)
tunnel source-port-hash: 5-tuple (default)
tunnel mode: vxlan-dummy-mac
tunnel mtu: 1450 (default)
tunnel source interface: Loopback0
dummy-mac-source: 0000.5e00.5213 (default)
dst-port: 4789 (default)
persistent: yes
```

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

Displays VxLAN static route configurations.

Example:

The output indicates the route profile associated with each prefix. In the following example, the 2.2.2.2/32 prefix is using the route profile called “test”.

```
Device# show vxlan static-route all
vrf test, topoid 2, IPv4
--------------------------------------------------
vrf test 2.2.2.2/32 vni 2 20.1.1.1 route-profile test persistent
vrf test 2.2.3.3/32 vni 5 20::1 route-profile test persistent
vrf test2, topoid 3, IPv4
--------------------------------------------------
vrf test2 2.2.2.5/32 vni 6 20::1 route-profile test2 persistent
vrf test2 2.2.2.6/32 vni 7 20::1 route-profile test2 persistent
vrf test2 2.2.2.8/32 vni 8 3.3.3.2 route-profile test2
vrf test2 2.2.2.8/32 vni 8 3.3.3.3 route-profile test2
vrf test2 2.2.2.8/32 vni 8 3.3.3.3 dst-mac 1212.1212.1212 route-profile test2
```

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, 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, and then displays details for the route with a bind-label of 0x2000002 (bold added in the output to highlight 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
```