Segmentation

Network segmentation has existed for over a decade and has been implemented in multiple forms and shapes. At its most rudimentary level, segmentation provides traffic isolation. The most common forms of network segmentation are virtual LANs, or VLANs, for Layer 2 solutions, and virtual routing and forwarding, or VRF, for Layer 3 solutions.

There are many use cases for segmentation:

Use Cases for Segmentation

- An enterprise wants to keep different lines of business separate (for example, for security or audit reasons).
- The IT department wants to keep authenticated users separate from guest users.
- A retail store wants to separate video surveillance traffic from transactional traffic.
- An enterprise wants to give business partners selective access only to some portions of the network.
- A service or business needs to enforce regulatory compliance, such as compliance with HIPAA, the U.S. Health Insurance Portability and Accountability Act, or with the Payment Card Industry (PCI) security standards.
- A service provider wants to provide VPN services to its medium-sized enterprises.
- An enterprise wants to set up a trial run of new service and wants to use a cloud service for development and system test.

Limitations of Segmentation

One inherent limitation of segmentation is its scope. Segmentation solutions either are complex or are limited to a single device or pair of devices connected using an interface. As an example, Layer 3 segmentation provides the following:

1. Ability to group prefixes into a unique route table (RIB or FIB).
2. Ability to associate an interface with a route table so that traffic traversing the interface is routed based on prefixes in that route table.

This is a useful functionality, but its scope is limited to a single device. To extend the functionality throughout the network, the segmentation information needs to be carried to the relevant points in the network.

How to Enable Network-Wide Segmentation

There are two approaches to providing this network-wide segmentation:

- Define the grouping policy at every device and on every link in the network (basically, you perform Steps 1 and 2 above on every device).
- Define the grouping policy at the edges of the segment, and then carry the segmentation information in the packets for intermediate nodes to handle.

The first approach is useful if every device is an entry or exit point for the segment, which is generally not the case in medium and large networks. The second approach is much more scalable and keeps the transport network free of segments and complexity. MPLS-based Layer 3 VPNs are a popular example of segmentation at the edge.

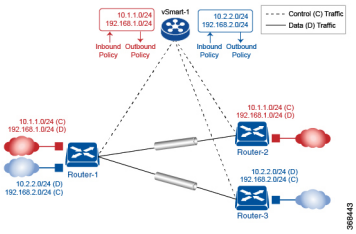
Segmentation in Cisco SD-WAN

In the Cisco SD-WAN overlay network, VPNs divide the network into different segments.

Cisco SD-WAN employs the more prevalent and scalable model of creating segments. Essentially, segmentation is done at the edges of a router, and the segmentation information is carried in the packets in the form of an identifier.

The figure shows the propagation of routing information inside a VPN .

Figure 1: Propagation of Routing Information Inside a VPN



In this figure:

- Router-1 subscribes to two VPNs , red and blue.
  - The red VPN caters to the prefix 10.1.1.0/24 (either directly through a connected interface or learned using the IGP or BGP).
  - The blue VPN caters to the prefix 10.2.2.0/24 (either directly through a connected interface or learned using the IGP or BGP).
- Router-2 subscribes to the red VPN .

- This VPN caters to the prefix 192.168.1.0/24 (either directly through a connected interface or learned using the IGP or BGP).

- Router-3 subscribes to the blue VPN .
  - This VPN caters to the prefix 192.168.2.0/24 (either directly through a connected interface or learned using the IGP or BGP).

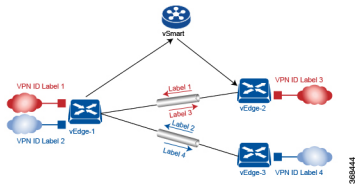
Because each router has an Overlay Management Protocol (OMP) connection over a TLS tunnel to a Cisco vSmart Controller, it propagates its routing information to the Cisco vSmart Controller. On the Cisco vSmart Controller, the network administrator can enforce policies to drop routes, to change TLOCs, which are overlay next hops, for traffic engineering or service chaining, or to change the VPN ID (see Policy Overview for more details). A network administrator can apply these policies as inbound and outbound policies on the Cisco vSmart Controller.

All the prefixes belonging to a single VPN are kept in a separate route table. This provides the Layer 3 isolation required for the various segments in the network. So, Router-1 has two VPN route tables, and Router-2 and Router-3 each have one route table. In addition, the Cisco vSmart Controller maintains the VPN context of each prefix.

Separate route tables provide isolation on a single node. So how is routing information propagated across the network?

In the Cisco SD-WAN solution, this is done using VPN identifiers, as shown in the figure below. AVPN ID, which is carried in a packet, identifies each VPN on a link. When you configure a VPN on a router, the VPN has a label associated with it. The router sends the label, along with the VPN ID, to the Cisco vSmart Controller. The Cisco vSmart Controller propagates this router-to-VPN ID mapping information to the other routers in the domain. The remote routers then use this label to send traffic to the appropriate VPN . The local routers, on receiving the data with the VPN ID label, use the label to demultiplex the data traffic. This is similar to how MPLS labels are used. This design is based on standard RFCs and is compliant with regulatory procedures such as PCI and HIPAA.

Figure 2: VPN Identifiers



Note The transport network that connects the routers is completely unaware of the VPNs . Only the routers know about VPNs ; the rest of the network follows standard IP routing.

VPNs Used in Cisco SD-WAN Segmentation

The Cisco SD-WAN solution provides two default VPNs to separate traffic: Transport VPN and Management VPN.

Transport VPNs

VPN 0 is the transport VPN. To enforce the inherent separation between services (such as prefixes that belong to the enterprise) and transport (the network that connects the vEdge routers), all the transport interfaces (that is, all the TLOCs) are kept in the transport VPN. This ensures that the transport network cannot reach the service network by default. Multiple transport interfaces can belong to the same transport VPN, and packets can be forwarded to and from transport interfaces. VPN 0 or transport VPN carries control traffic over secure DTLS or TLS connections between vSmart controllers and vEdge routers, and between vSmart controllers and vBond orchestrators

VPN 0 contains all interfaces for a device except for the management interface, and all the interfaces are disabled. For the control plane to establish itself so that the overlay network can function, you must configure WAN transport interfaces in VPN 0. On vEdge routers, the interfaces in VPN 0 connect to some type of transport network or cloud, such as the Internet, MPLS, or Metro Ethernet. For each interface in VPN 0, you must set an IP address, and you create a tunnel connection that sets the color and encapsulation for the WAN transport connection. (The encapsulation is used for the transmission of data traffic.) These three parameters—IP address, color, and encapsulation—define a TLOC (transport location) on the vEdge router. The OMP session running on each tunnel sends the TLOC to the vSmart controllers so that they can learn the overlay network topology. For VPN 0, you can also set other interface-specific and VPN-specific properties in VPN 0.

Dual Stack Support on Transport VPNs

In the transport VPN (VPN 0), vEdge routers and vSmart controllers support dual stack. To enable dual stack, configure an IPv4 address and an IPv6 address on the tunnel interface. The vEdge router learns from the vSmart controller whether a destination supports IPv4 or IPv6 addresses. When forwarding traffic, the router chooses either the IPv4 or the IPv6 TLOC based on the destination address.

Management VPNs

VPN 512 is the management VPN. It carries out-of-band network management traffic among the Cisco SD-WAN devices in the overlay network. By default, VPN 512 is configured and enabled. You can modify this configuration if desired.

Service VPNs

To segment user networks and user data traffic locally at each site and to interconnect user sites across the overlay network, you create additional VPNs on Cisco vEdge devices. These VPNs are identified by a number that is not 0 or 512. To enable the flow of data traffic, you associate interfaces with each VPN, assigning an IP address to each interface. These interfaces connect to local-site networks, not to WAN transport clouds. For each of these VPNs, you can set other interface-specific properties, and you can configure features specific for the user segment, such as BGP and OSPF routing, VRRP, QoS, traffic shaping, and policing.

Configure VPNs Using Cisco vManage Templates

Create a VPN Template

Note You can configure a static route through the VPN template.

Step 1 From the Cisco vManage menu, choose **Configuration** > **Templates**.

Step 2 Click **Device Templates**, and click **Create Template**.

Note In Cisco vManage Release 20.7.x and earlier releases **Device Templates** is called **Device**.

Step 3 From the **Create Template** drop-down list, choose **From Feature Template**.

Step 4 From the **Device Model** drop-down list, choose the type of device for which you wish to create the template.

Step 5 To create a template for VPN 0 or VPN 512:

a. Click **Transport & Management VPN**, or scroll to the **Transport & Management VPN** section.

b. From the VPN 0 or VPN 512 drop-down list, click **Create Template**. The VPN template form appears.

The form contains fields for naming the template, and fields for defining VPN parameters.

Step 6 To create a template for VPNs 1 through 511, and 513 through 65530:

a. Click **Service VPN**, or scroll to the **Service VPN** section.

b. Click the **Service VPN** drop-down list.

c. From the **VPN** drop-down list, click **Create Template**. The VPN template form displays.

The form contains fields for naming the template, and fields for defining VPN parameters.

Step 7 In **Template Name**, enter a name for the template. The name can be up to 128 characters and can contain only alphanumeric characters.

Step 8 In **Template Description**, enter a description of the template. The description can be up to 2048 characters and can contain only alphanumeric characters.

Configure Basic VPN Parameters

To configure basic VPN parameters, choose **Basic Configuration** and then configure the following parameters. Parameters marked with an asterisk are required to configure a VPN.

Parameter Name	Description
VPN	Enter the numeric identifier of the VPN. Range for Cisco vEdge devices: 0 through 65530 Values for Cisco vSmart Controller and Cisco vManage devices: 0, 512
Name	Enter a name for the VPN.
Enhance ECMP keying	Click **On** to enable the use in the ECMP hash key of Layer 4 source and destination ports, in addition to the combination of the source IP address, destination IP address, protocol, and DSCP field, as the ECMP hash key. ECMP keying is **Off** by default.

Parameter Name	Description
Enable TCP Optimization Cisco vEdge devices only	Click **On** to enable TCP optimization for a service-side VPN (a VPN other than VPN 0 and VPN 512). TCP optimization fine-tunes TCP to decrease round-trip latency and improve throughput for TCP traffic.

Note To complete the configuration of the transport VPN on a router, you must configure at least one interface in VPN 0.

To save the feature template, click **Save**.

Configure Basic Interface Functionality

To configure basic interface functionality in a VPN, choose **Basic Configuration** and configure the following parameters:

Note Parameters marked with an asterisk are required to configure an interface.

Parameter Name	IPv4 or IPv6	Options	Description
Shutdown*	Click **No** to enable the interface.		
Interface name*	Enter a name for the interface.		
Description	Enter a description for the interface.		
IPv4 / IPv6	Click **IPv4** to configure an IPv4 VPN interface. Click **IPv6** to configure an IPv6 interface.		
Dynamic	Click **Dynamic** to set the interface as a Dynamic Host Configuration Protocol (DHCP) client, so that the interface receives its IP address from a DHCP server.		
	Both	**DHCP Distance**	Optionally, enter an administrative distance value for routes learned from a DHCP server. Default is 1.
	IPv6	**DHCP Rapid Commit**	Optionally, configure the DHCP IPv6 local server to support DHCP Rapid Commit, to enable faster client configuration and confirmation in busy environments. Click **On** to enable DHCP rapid commit. Click **Off** to continue using the regular commit process.
Static	Click **Static** to enter an IP address that doesn't change.		
	IPv4	**IPv4 Address**	Enter a static IPv4 address.
	IPv6	**IPv6 Address**	Enter a static IPv6 address.

Parameter Name	IPv4 or IPv6	Options	Description
Secondary IP Address	**IPv4**	Click **Add** to enter up to four secondary IPv4 addresses for a service-side interface.	
IPv6 Address	**IPv6**	Click **Add** to enter up to two secondary IPv6 addresses for a service-side interface.	
DHCP Helper	**Both**	To designate the interface as a DHCP helper on a router, enter up to eight IP addresses, separated by commas, for DHCP servers in the network. A DHCP helper interface forwards BootP (broadcast) DHCP requests that it receives from the specified DHCP servers.	
Block Non-Source IP	**Yes** / **No**	Click **Yes** to have the interface forward traffic only if the source IP address of the traffic matches the interface's IP prefix range. Click **No** to allow other traffic.	
Bandwidth Upstream	For Cisco vEdge devices and vManage: For transmitted traffic, set the bandwidth above which to generate notifications. Range: 1 through (232 / 2) – 1 kbps		
Bandwidth Downstream	For Cisco vEdge devices and vManage: For received traffic, set the bandwidth above which to generate notifications. Range: 1 through (232 / 2) – 1 kbps		

To save the feature template, click **Save**.

CLI Equivalent

```
vpn vpn-id
  interface interface-name
    bandwidth-downstream kbps
    bandwidth-upstream kbps
    block-non-source-ip
    description text
    dhcp-helper ip-address
    (ip address ipv4-prefix/length| ip dhcp-client [dhcp-distance number])
    (ipv6 address ipv6-prefix/length | ipv6 dhcp-client [dhcp-distance number]
[dhcp-rapid-commit])
    secondary-address ipv4-address
    [no] shutdown
```

Create a Tunnel Interface

On Cisco vEdge device s, you can configure up to eight tunnel interfaces. This means that each Cisco vEdge device router can have up to eight TLOCs. On Cisco vSmart Controllers and Cisco vManage, you can configure one tunnel interface.

For the control plane to establish itself so that the overlay network can function, you must configure WAN transport interfaces in VPN 0. The WAN interface will enable the flow of tunnel traffic to the overlay. You can add other parameters shown in the table below only after you configure the WAN interface as a tunnel interface.

To configure a tunnel interface, select **Interface Tunnel** and configure the following parameters:

To configure additional tunnel interface parameters, click **Advanced Options**:

Parameter Name	Cisco vEdge devices Only	Description
GRE	Yes	Use GRE encapsulation on the tunnel interface. By default, GRE is disabled. If you select both IPsec and GRE encapsulations, two TLOCs are created for the tunnel interface that have the same IP addresses and colors, but that differ by their encapsulation.
IPsec	Yes	Use IPsec encapsulation on the tunnel interface. By default, IPsec is enabled. If you select both IPsec and GRE encapsulations, two TLOCs are created for the tunnel interface that have the same IP addresses and colors, but that differ by their encapsulation.
IPsec Preference	Yes	Specify a preference value for directing traffic to the tunnel. A higher value is preferred over a lower value. Range: 0 through 4294967295 Default: 0
IPsec Weight	Yes	Enter a weight to use to balance traffic across multiple TLOCs. A higher value sends more traffic to the tunnel. Range: 1 through 255 Default: 1
Carrier	No	Select the carrier name or private network identifier to associate with the tunnel. Values: carrier1, carrier2, carrier3, carrier4, carrier5, carrier6, carrier7, carrier8, default Default: default
Bind Loopback Tunnel	Yes	Enter the name of a physical interface to bind to a loopback interface.
Last-Resort Circuit	Yes	Select to use the tunnel interface as the circuit of last resort.
NAT Refresh Interval	No	Enter the interval between NAT refresh packets sent on a DTLS or TLS WAN transport connection. Range: 1 through 60 seconds Default: 5 seconds

Parameter Name	Cisco vEdge devices Only	Description
Hello Interval	No	Enter the interval between Hello packets sent on a DTLS or TLS WAN transport connection. Range: 100 through 10000 milliseconds Default: 1000 milliseconds (1 second)
Hello Tolerance	No	Enter the time to wait for a Hello packet on a DTLS or TLS WAN transport connection before declaring that transport tunnel to be down. Range: 12 through 60 seconds Default: 12 seconds

Configure DNS and Static Hostname Mapping

To configure DNS addresses and static hostname mapping, click **DNS** and configure the following parameters:

Parameter Name	Options	Description
Primary DNS Address	Click either **IPv4** or **IPv6**, and enter the IP address of the primary DNS server in this VPN.	
New DNS Address	Click **New DNS Address** and enter the IP address of a secondary DNS server in this VPN. This field appears only if you have specified a primary DNS address.	
	Mark as Optional Row	Check the **Mark as Optional Row** check box to mark this configuration as device-specific. To include this configuration for a device, enter the requested variable values when you attach a device template to a device, or create a template variables spreadsheet to apply the variables.
	Hostname	Enter the hostname of the DNS server. The name can be up to 128 characters.
	List of IP Addresses	Enter up to eight IP addresses to associate with the hostname. Separate the entries with commas.
To save the DNS server configuration, click **Add**.		

To save the feature template, click **Save**.

CLI Equivalent

```
vpn vpn-id
  dns ip-address (primary | secondary)
  host hostname ip ip-address
```

Configure Segmentation Using the CLI

Configure VPNs Using the CLI

Configure Transport VPN on a vEdge Router

On a vEdge router, the interfaces in VPN 0 connect to a WAN transport network. You must configure at least one tunnel interface on a vEdge router so that it can join the control plane and be part of the overlay network. If is not configured, that router cannot participate in the overlay network.

For a tunnel connection on a vEdge router, you must configure the three components of a TLOC—the interface's IP address and the tunnel's color and encapsulation. An OMP session runs over each tunnel connection, and it is OMP that distributes the device TLOCs to vSmart controllers. The controllers use the TLOCs to determine the overlay network topology and to determine the best routing paths across the overlay network. A vEdge router can have up to four TLOCs, so you can configure more than one tunnel connection.

In the transport VPN (VPN 0), vEdge routers support dual stack. To enable dual stack, configure an IPv4 address and an IPv6 address on the tunnel interface. The vEdge router learns from the vSmart controller whether a destination supports IPv4 or IPv6 addresses. When forwarding traffic, the router chooses either the IPv4 or the IPv6 TLOC based on the destination address.

To configure VPN 0 on a vEdge router:

1. Configure the WAN transport interface:

   ```
   vEdge(config)# vpn 0 interface interface-name
   vEdge(config-interface)#
   ```

   In the most common cases, *interface-name* is the name of a physical Gigabit Ethernet interface (**ge** *port* **/** *slot*). The interface name can also be **gre** *number*, **ipsec** *number*, **loopback** *string*, **natpool** *number*, or **ppp** *number*.

2. Configure a static IPv4 address for the interface:

   ```
   vEdge(config-interface)# ip address prefix/length
   vEdge(config-interface) #
   ```

   Or you can enable DHCP on the interface so that the interface learn its IP address dynamically:

   ```
   vEdge(config-interface)# ip dhcp-client [dhcp-distance number]
   vEdge(config-interface)#
   ```

   When an interface learns its IPv4 address from a DHCP server, it can also learn routes from the server. By default, these routes have an administrative distance of 1, which is the same as static routes. To change the default value, include the **dhcp-distance** option, specifying a distance from 1 through 255.

3. To enable dual stack, configure a static IPv6 address for the interface:

   ```
   vEdge(config-interface)# ipv6 address prefix/length
   vEdge(config-interface)#
   ```

   Or you can enable DHCPv6 on the interface so that the interface learn its IP address dynamically:

   ```
   vEdge(config-interface)# ipv6 dhcp-client [dhcp-distance number] [dhcp-rapid-commit]
   vEdge(config-interface)#
   ```

   When an interface learns its IPv6 address from a DHCPv6 server, it can also learn routes from the server. By default, these routes have an administrative distance of 1, which is the same as static routes. To

change the default value, include the **dhcp-distance** option, specifying a distance from 1 through 255. To speed up the assignment of IPv6 addresses, include the **dhcp-rapid-commit** option.

4. Enable the interface:

```
vEdge(config-interface)# no shutdown
```

5. Configure the WAN transport tunnel connection:

```
vEdge(config-interface)# tunnel-interface
vEdge(config-tunnel-interface)#
```

6. Configure a color for the tunnel connection as an identifier for the tunnel:

```
vEdge(config-tunnel-interface)# color color
vEdge(config-tunnel-interface)#
```

color can be **3g**, **biz-internet**, **blue**, **bronze**, **custom1**, **custom2**, **custom3**, **default**, **gold**, **green**, **lte**, **metro-ethernet**, **mpls**, **private1** through **private6**, **public-internet**, **red**, and **silver**. The default color is **default**. The colors **metro-ethernet**, **mpls**, and **private1** through **private6** are referred to as *private colors*, because they use private addresses to connect to the remote side vEdge router in a private network. You can use these colors in a public network provided that there is no NAT device between the local and remote vEdge routers.

7. Configure the encapsulation to use on tunnel connection:

```
vEdge(config-tunnel-interface)# encapsulation (gre | ipsec)
vEdge(config-tunnel-interface)#
```

To configure both IPsec and GRE encapsulation, include two **encapsulation** commands. Note that if you do this, you are creating two TLOCs that have the same IP addresses and colors, but that have different encapsulation.

8. Configure any other properties specific to the tunnel interface, the interface, or VPN 0.

9. If you have a multi-TLOC environment, configure additional tunnel interfaces.

10. Enable DNS service in the VPN by configuring the IP address of a DNS server reachable from VPN 0:

```
vEdge(config-vpn-0)# dns ip-address (primary | secondary)
```

The address can be either an IPv4 or IPv6 address. By default, the IP address is for the primary DNS server.

11. If desired, configure IPv4 and IPv6 static routes in VPN 0:

```
vEdge(config-vpn-0)# ip route prefix/length next-hop [administrative-distance]
vEdge(config-vpn-0)# ipv6 route prefix/length next-hop [administrative-distance]
```

12. Activate the configuration:

```
vEdge(config)# commit
```

To display interface information, use the **show interface** command for IPv4 interfaces and **show ipv6 interfaces** for IPv6 interfaces. To display information about DHCP and DHCPv6 servers, use the **show dhcp interface** and **show ipv6 dhcp interface** commands.

When you are troubleshooting routing and forwarding problems on a vEdge router, you can configure the router to perform route consistency checks, to determine whether the routes in the router's route and forwarding tables are consistent:

```
vEdge(config-system)#route-consistency-check
```

This command checks only IPv4 routes. Route consistency checking requires a large amount of device CPU, so it is recommended that you enable it only when you trouble shooting an issue and that you disable it at other times.

Here is an example of a VPN 0 configuration, where **interface ge0/0** is the WAN transport interface. This example shows that dual stack is enabled on the router, because the tunnel interface has both an IPv4 and an IPv6 address. Notice that the remaining seven device interfaces are part of VPN 0, because we have not yet configured any other VPNs. Also notice that the management interface is not present in VPN 0.

```
vpn 0
 interface ge0/0
  ip address 10.0.0.8/24
  ipv6 address fd00:1234::/16
  tunnel-interface
   color biz-internet
   encapslation ipsec
   allow-service dhcp
   allow-service dns
   allow-service icmp
   no allow-service sshd
   no allow-service ntp
   no allow-service stun
  !
  no shutdown
 !
 interface ge0/1
  shutdown
 !
 interface ge0/2
  shutdown
 !
 interface ge0/3
  shutdown
 !
 interface ge0/4
  shutdown
 !
 interface ge0/5
  shutdown
 !
 interface ge0/6
  shutdown
 !
 interface ge0/7
  shutdown
 !
!
```

An interface can participate only in one VPN. So in an initial configuration, when VPN 0 is the only VPN that is configured, all the device's interfaces are present, by default, in VPN 0 (as shown in the output above). Then, when you create other VPNs to carry data traffic and configure interfaces in those VPNs, the interfaces used in the other VPNs are automatically removed from VPN 0. Here is an example in which **interface ge0/3** is used for VPN 1, so it has been automatically removed from the configuration of VPN 0:

```
vpn 0
interface ge0/0
  ip address 10.0.0.8/24
  tunnel-interface
   color biz-internet
   encapsulation ipsec
   allow-service dhcp
   allow-service dns
   allow-service icmp
```

```
   no allow-service sshd
   no allow-service ntp
   no allow-service stun
  !
  no shutdown
!
interface ge0/1
  shutdown
!
interface ge0/2
  shutdown
!
interface ge0/4
  shutdown
!
interface ge0/5
  shutdown
!
interface ge0/6
  shutdown
!
interface ge0/7
  shutdown
!
!
vpn 1
router
  ospf
   redistribute omp route-policy test-policy
   area 0
    interface ge0/3
    exit
   exit
  !
!
interface ge0/3
  ip address 10.10.10.1/24
  no shutdown
!
!
```

When you configure subinterfaces in a VPN that carries data traffic (that is, not VPN 0 and not VPN 512), the main interface must be configured with the **no shutdown** command so that it is enabled, and the main interface remains in VPN 0 once you configure the subinterface. For example, if in the VPN 1 configuration, you were to configure OSPF on VLAN 1, you can see that **interface ge0/3** remains present in VPN 0, while the subinterface **interface ge0/3.1** is used in VPN1:

```
vpn 0
 dns 1.2.3.4 primary
 interface ge0/0
  address 10.0.0.8/24
  tunnel-interface
   preference 100
   allow-service dhcp
   allow-service dns
   allow-service icmp
   allow-service sshd
   allow-service ntp
   allow-service stun
  !
  no shutdown
 !
 interface ge0/1
  shutdown
```

```
 !
 interface ge0/2
  shutdown
 !
 interface ge0/3
  no shutdown
 !
 interface ge0/4
  shutdown
 !
 interface ge0/5
  shutdown
 !
 interface ge0/6
  shutdown
 !
 interface ge0/7
  shutdown
 !
!
vpn 1
router
  ospf
   redistribute omp route-policy test-policy
   area 0
    interface ge0/3.1
    exit
   exit
  !
 !
interface ge0/3.1
  ip address 10.10.10.1/24
  no shutdown
 !
!
```

Configure the Transport VPN on a vSmart Controller

Because vSmart controllers are responsible for determining the best routes through the overlay network (based on the TLOCs it learns and based on centralized policies), they handle only control plane traffic, in VPN 0. A vSmart controller can have only one interface in VPN 0, for which you set an IP address and you create a tunnel connection. This tunnel connection acts a control plane tunnel termination point.

In the transport VPN (VPN 0), vEdge routers support dual stack. To enable dual stack, configure an IPv4 address and an IPv6 address on the tunnel interface. The vEdge router learns from the vSmart controller whether a destination supports IPv4 or IPv6 addresses. When forwarding traffic, the router chooses either the IPv4 or the IPv6 TLOC based on the destination address.

To configure VPN 0 on a vSmart controller:

1. Configure the WAN transport interface:

   ```
   vSmart(config)# vpn 0 interface interface-name
   vSmart(config-interface)#
   ```

   interface-name is the name of a virtual Ethernet interface (**eth** *number*).

2. Configure a static IPv4 address for the interface:

   ```
   vSmart(config-interface)# ip address prefix/length
   vSmart(config-interface)#
   ```

   Or you can enable DHCP on the interface so that the interface learn its IP address dynamically:

```
vSmart(config-interface)# ip dhcp-client [dhcp-distancenumber]
vSmart(config-interface)#
```

When an interface learns its IPv4 address from a DHCP server, it can also learn routes from the server. By default, these routes have an administrative distance of 1, which is the same as static routes. To change the default value, include the **dhcp-distance** option, specifying a distance from 1 through 255.

3. To enable dual stack, configure a static Pv6 address for the interface:

```
vSmart(config-interface)# ipv6 address prefix/length
vSmart(config-interface)#
```

Or you can enable DHCPv6 on the interface so that the interface learn its IP address dynamically:

```
vSmart(config-interface)# ipv6 dhcp-client [dhcp-distance number] [dhcp-rapid-commit]
vSmart(config-interface)#
```

When an interface learns its IPv6 address from a DHCPv6 server, it can also learn routes from the server. By default, these routes have an administrative distance of 1, which is the same as static routes. To change the default value, include the **dhcp-distance** option, specifying a distance from 1 through 255. To speed up the assignment of IPv6 addresses, include the **dhcp-rapid-commit** option.

4. Enable the interface:

```
vSmart(config-interface)# no shutdown
```

5. Enable DNS service in the VPN by configuring the IP address of a DNS server reachable from VPN 0:

```
vSmart(config-vpn-0)# dns ip-address (primary | secondary)
```

The address can be either an IPv4 or IPv6 address. By default, the IP address is for the primary DNS server.

6. If desired, configure IPv4 and IPv6 static routes in VPN 0:

```
vSmart(config-vpn-0)# ip route prefix/length next-hop [administrative-distance]
vSmart(config-vpn-0)# ipv6 route prefix/length next-hop [administrative-distance]
```

7. Configure any other properties specific to the tunnel interface, the interface, or VPN 0.

8. Activate the configuration:

```
vSmart(config)# commit
```

To display interface information, use the **show interface** command for IPv4 interfaces and **show ipv6 interfaces** for IPv6 interfaces. To display information about DHCP and DHCPv6 servers, use the **show dhcp interface** and **show ipv6 dhcp interface** commands.

Here is an example of a VPN 0 configuration on a vSmart controller:

```
vSmart# show running-config vpn 0
vpn 0
 dns 1.2.3.4 primary
 interface eth0
  ip dhcp-client
  no shutdown
 !
 interface eth1
  ip address 10.0.5.19/24
  tunnel-interface
   allow-ssh
   allow-icmp
  !
  no shutdown
 !
```

```
 ip route 0.0.0.0/0 10.0.5.13
!
```

Configure Data Traffic Exchange across Private WANs

When a vEdge router is connected to a private WAN, such as an MPLS or a metro Ethernet network, the carrier hosting the private network does not advertise the IP address of that vEdge router over the internet. (This IP address is associated with the TLOC on that vEdge router.) This means that remote vEdge routers are not able to learn how to reach that router and hence are not able to exchange data traffic with it directly over the private network.

To allow the vEdge router behind the private network to communicate directly over the private WAN with other vEdge routers, you direct the data traffic to a loopback interface rather than to the actual physical WAN interface. The overlay network can then advertise that the local router is reachable via its loopback address. To make it possible for the data traffic to actually be transmitted out the WAN interface, you bind the loopback interface to the physical WAN interface to the private network.

To configure VPN 0 so that it carries data traffic across private WANs:

1. Configure the loopback interface, assigning it an IP address:

   ```
   vEdge(config)# vpn 0  loopback
   number ip address prefix/length
   vEdge(config-loopback)# no shutdown
   ```

2. Configure the loopback interface to be a transport interface:

   ```
   vEdge(config-loopback)# tunnel-interface
   ```

3. Set the color of the loopback interface to be one of the primatel colors—**metro-ethernet**, **mpls**, and **private1** through **private6**. You must configure this same color on the loopback interfaces of all vEdge routers in the same private LAN.

   ```
   vEdge(config-tunnel-interface)# color color
   ```

Use the **show interface** command to check that the loopback interface in configured properly, as a transport interface with the proper IP address and color.

If a single vEdge router is connected to two (or more) different private networks, create a loopback interface for each private network, associate a carrier name with the interface so that the router can distinguish between the two private WANs, and "bind" the loopback interface to the physical interface that connects to the appropriate private WAN:

1. Configure the loopback interface, assigning it an IP address:

   ```
   vEdge(config)# vpn 0
   loopback
   number
   ip address prefix/length
   vEdge(config-loopback)# no shutdown
   ```

2. Configure the loopback interface to be a transport interface and bind it to a physical interface:

   ```
   vEdge(config-loopback)# tunnel-interface bind
   ge
   slot/port
   ```

3. Configure a carrier name and TLOC color on the loopback interface:

   ```
   vEdge(config-tunnel-interface)# carrier carrier-name
   vEdge(config-tunnel-interface)# color color
   ```

4. On the physical interface, configure its IP address, and enable it:

```
vEdge(config)# vpn 0 interface
ge
slot/port
ip address prefix/length
vEdge(config-ge)# no shutdown
```

Configure the Management VPN (VPN 512)

In the Cisco SD-WAN overlay network, VPN 512 is the network management VPN. It carries out-of-band management traffic in the overlay network. VPN 512 is configured and enabled by default on all Cisco SD-WAN devices. It contains the interface used for management traffic. For vEdge routers, this interface is generally a Gigabit Ethernet (**ge**) interface, and for other Cisco SD-WAN devices it is an **eth** interface. DHCP is enabled by default on the management interface. The default configuration for VPN 512 on a vEdge router looks like this:

```
vpn 512
 interface ge0/0
  ip dhcp-client
  no shutdown
 !
!
```

VPN 512 must be present on all Cisco SD-WAN devices so that they are always reachable on the network. You can configure additional parameters for VPN 512 if you choose.

Configure VPNs To Carry Data Traffic

VPNs other than VPN 0 and VPN 512 are used to carry data traffic across the overlay network. These VPNs are sometimes referred to as *service-side VPNs*. For these VPNs to operate, each one must have an operational interface (or subinterface). The remainder of what you configure in these VPNs depends on your network needs. You configure features specific for the user segment, such as BGP and OSPF routing, VRRP, QoS, traffic shaping, and policing.

To create a data traffic VPN:

1. Configure the VPN:

```
vEdge(config)# vpn
number
vEdge(config-vpn)#
```

   The VPN number can be in the range 1 through 511, and 513 through 65535.

2. Configure at least one interface in the VPN and its IP address:

```
vEdge(config-vpn)# interface
 interface-name
 ip address
 address/prefix
vEdge(config-interface)#
```

   The interface name has the format **ge** *slot*/*port*, where the slot is generally 0 through 7 (depending on the device) and the port is 0 through 8. If you are configuring VLANs, specify a subinterface name in the format **ge** *slot*/*port* **.** *vlan*, where the VLAN number can be in the range 1 through 4094. (VLAN numbers 0 and 4095 are reserved.) The interface name can also be **gre** *number*, **ipsec** *number*, **loopback** *string*, **natpool** *number*, or **ppp** *number*.

3. Activate the interface:

```
vEdge(config-interface)# no shutdown
```

4. Enable DNS service in the VPN by configuring the IP address of a DNS server reachable from that VPN:

```
vEdge(config-vpn)#  dns ip-address
```

5. If desired, configure IPv4 static routes in the VPN:

```
vEdge(config-vpn)# ip route  prefix
/
length next-hop [administrative-distance]
```

6. Configure any other properties specific to the interface or to VPN.

7. Activate the configuration:

```
vEdge(config)# commit
```

Here is an example of a configuration for VPN 1:

```
vpn 1
 dns 1.2.3.4 primary
 router
  ospf
   redistribute omp route-policy test-policy
   area 0
    interface ge0/3
    exit
   exit
  !
 !
 interface ge0/3
  ip address 10.10.10.1/24
  no shutdown
 !
!
```

Dual-Stack Operation

When a Cisco SD-WAN device establishes an IPsec tunnel for control traffic between a local TLOC and a remote TLOC, or when a device establishes a BFD tunnel for data plane traffic between a local and a remote TLOC, an IPv6 tunnel is established in the following situations:

- The local device has only an IPv6 address, and the remote device has an IPv6 address.
- The remote device has only an IPv6 address, and the local device has an IPv6 address.

If both the local and remote devices have IPv4 addresses, IPsec and BFD always establish an IPv4 tunnel.

Segmentation (VPNs) Configuration Examples

Some straightforward examples of creating and configuring VPNs to help you understand the configuration procedure for segmenting networks.

Create Basic VPNs

Creating the basic VPNs required by Cisco SD-WAN devices is a simple, straightforward process, consisting of these steps:

1. On the vEdge router:

- Create a VPN instance for the transport VPN. VPN 0 is reserved for the transport VPN.
- Create a VPN instance for the management VPN. VPN 512 is reserved for the management VPN.
- Create a VPN instance to use for routing.

2. On the vSmart controller:
   - Create a VPN instance for the transport VPN. VPN 0 is reserved for the transport VPN.
   - Create a VPN instance for the management VPN. VPN 512 is reserved for the management VPN.
   - Optionally, create policies to influence routing and access control within the VPN.

Configuration on the vEdge Router

To create the basic VPNs on a vEdge router, you configure VPN 0 for transport, VPN 512 for management, and a third VPN (here, VPN 1) for carrying data traffic:

1. First, configure general system parameters:

```
vEdge(config)# system host-name host-name
vEdge(config-system)# system-ip ip-address
vEdge(config-system)# domain-id domain-id
vEdge(config-system)# site-id site-id
vEdge(config-system)# vbond (dns-name | ip-address)
```

2. In VPN 0, which is the transport VPN, configure the interface to the WAN transport cloud, to establish reachability between the vEdge router and the vSmart controller, and between vEdge routers:
   a. Configure an IP address for the interface:

```
vEdge(config-interface)# vpn 0 interface interface-name ip address prefix/length
```

   b. Enable the interface:

```
vEdge(config-interface)# no shutdown
```

   c. Enable a transport tunnel interface to carry control and data traffic, and configure the color and encapsulation for the tunnel:

```
vEdge(config-interface)# tunnel-interface
vEdge(config-tunnel-interface)# encapsulation (gre | ipsec)
vEdge(config-tunnel-interface)# color color
```

   d. Configure a default route for the VPN:

```
vEdge(config-vpn-0)# ip route 0.0.0.0/0 ip-address
```

3. Configure a VPN for data traffic:
   a. Create the VPN and assign it a identifier number. The identifier can be any number except 0 and 512.

```
vEdge(config)# vpn vpn-id
```

   b. Add an interface to the VPN:

```
vEdge(config-vpn-number)# interface interface-name ip address ip-address
```

   c. Enable the interface:

```
vEdge(config-vpn-number)# no shutdown
```

4. Configure unixAR routing in the VPN. See Configuring Basic Unicast Overlay Routing for more information.
5. Activate the configuration:

```
vEdge(config)# commit
```

Here is the full configuration on the vEdge router:

```
system                    # Configure general system parameters
 host-name vedge
 system-ip 1.0.0.2
 domain-id 1
 site-id   20
 vbond 10.2.6.1
!
vpn 0                               # Create the tunnel interface and allow
  interface ge 0/0                    reachability to vSmart in transport VPN
    ip address 10.2.6.11/24
    tunnel-interface
      color default
      encapsulation ipsec
   !
   no shutdown
   !
 ip route 0.0.0.0/0 10.2.6.12
!
vpn 1                               # Create new VPN, add interfaces and routing
 interface ge 0/1
  ip address 10.100.1.1/24
  no shutdown
 !
!
router
  bgp 20
   neighbor 10.100.1.2
    no shutdown
    remote-as 20
    address-family ipv4_unicast
    !
   !
  !
 !
vpn 512
  interface mgmt0
    ip dhcp-client
    no shutdown
  !
!
```

Configuration on the Cisco vSmart Controller

On the Cisco vSmart Controller, you configure general system parameters and the two VPNs—VPN 0 for WAN transport and VPN 512 for network management—as you did for the Cisco IOS XE SD-WAN device. Also, you generally create a centralized control policy that controls how the VPN traffic is propagated through the rest of the network. In this particular example, we create a central policy, shown below, to drop unwanted prefixes from propagating through the rest of the network. You can use a single Cisco vSmart Controller policy to enforce policies throughout the network.

Here are the steps for creating the control policy on the Cisco vSmart Controller:

1. Create a list of sites IDs for the sites where you want to drop unwanted prefixes:

```
vSmart(config)# policy lists site-list 20-30 site-id 20
vSmart(config-site-list-20-30)# site-id 30
```

2. Create a prefix list for the prefixes that you do not want to propagate:

```
vSmart(config)# policy lists prefix-list drop-list ip-prefix 10.200.1.0/24
```

3. Create the control policy:

```
vSmart(config)# policy control-policy drop-unwanted-routes sequence 10 match route
prefix-list drop-list
vSmart(config-match)# top
vSmart(config)# policy control-policy drop-unwanted-routes sequence 10 action reject
vSmart(config-action)# top
vSmart(config)# policy control-policy drop-unwanted-routes sequence 10 default-action
accept
vSmart(config-default-action)# top
```

4. Apply the policy to prefixes inbound to the Cisco vSmart Controller controller:

```
vSmart(config)# apply-policy site-list 20-30 control-policy drop-unwanted-routes in
```

Here is the full policy configuration on the Cisco vSmart Controller controller:

```
apply-policy
 site-list 20-30
  control-policy drop-unwanted-routes in
 !
!
policy
 lists
  site-list 20-30
   site-id 20
   site-id 30
  !
  prefix-list drop-list
   ip-prefix 10.200.1.0/24
  !
 !
 control-policy drop-unwanted-routes
  sequence 10
   match route
    prefix-list drop-list
   !
   action reject
   !
  !
  default-action accept
 !
!
```

Control VPN Membership

You can create VPNs just at the sites of interest and can then keep them hidden so that the rest of the network does not even know about them and the routes from them. Such a network design provides a great deal of traffic isolation and flexibility. However, there might be cases where the network administrator might want to explicitly disallow the creation of VPNs on the vEdge router. An example is in a B2B partnership, when the vEdge router is not located at the customer premise. For these situations, the network administrator can choose to allow only certain VPNs on these vEdge routers. Effectively, you are controlling membership in the VPN.

You control VPN membership policy at the vSmart controller. In the example here, you create a policy that explicitly disallows VPN 1 at sites 20 and 30:

```
apply-policy
 site-list 20-30
  vpn-membership disallow-vpn1
 !
!
policy
 lists
  site-list 20-30
   site-id 20
   site-id 30
  !
 !
 vpn-membership disallow-vpn1
  sequence 10
   match vpn-id 1
   action reject
   !
  !
  default-action accept
 !
!
```

Leak Routes across VPNs

In some situations it is desirable to leak routes from one VPN into another. Some examples include extranets, where you are making a portion of your intranet available to users outside your organization, B2B partnerships, and the network transition that occurs during a merger or acquisition. To leak routes across VPNs, you create a leaking control policy on the vSmart controller, a design that allows you to control route leaking from a central point in the network.

In this example, we create a control policy that allows an enterprise's VPN to import routes from a VPN list. Specifically, we:

- Create a control policy to match routes from a list of VPNs. Here, sequence 10 of the policy matches all routes from the VPNs of all business partners (BPs). The business partner VPN IDs are listed in the **All-BPs** list.
- Accept routes that match this policy, and import the prefixes into a new VPN called **Enterprise-BP**.
- Apply this policy towards the BP sites on vRoutes inbound to the vSmart controller.

```
policy
  lists
    site-list BP-Sites
      site-id 10
      site-id 20
    vpn-list All-BPs
      vpn 100
      vpn 101
    vpn-list Enterprise-BP
      vpn 200
  control-policy import-BPs-to-Enterprise
    sequence 10
     match route
      vpn-list All-BPs
     !
     action accept
      export-to vpn-list Enterprise-BP
      !
```

```
     !
    !
    default-action accept
   !
!
apply-policy
 site-list BP-Sites
  control-policy import-BPs-to-Enterprise in
 !
```

This policy matches all routes from all VPNs in the **All-BPs** VPN lists and populates these prefixes into the VPNs in the Enterprise-BP list. The routing table of the Enterprise-BP VPN will now contain all the prefixes of the BPs.

One advantage of importing routes in this way is access control. Keeping each BP in a separate VPN and creating an extranet policy ensures that the BPs cannot talk to each other.

Use Case: Exchange Data Traffic within a Single Private WAN

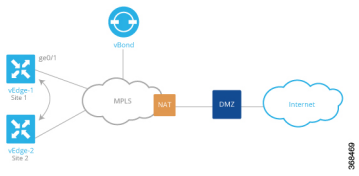
Allow Data Traffic Exchange across Private WANs

When the WAN network to which a vEdge router is connected is a private network, such as an MPLS or a metro Ethernet network, and when the carrier hosting the private network does not advertise the router's IP address, remote vEdge routers on the same private network but at different sites can never learn how to reach that router and hence are not able to exchange data traffic with it by going only through the private network. Instead, the remote routers must route data traffic through a local NAT and over the Internet to a vBond orchestrator, which then provides routing information to direct the traffic to its destination. This process can add significant overhead to data traffic exchange, because the vBond orchestrator may physically be located at a different site or a long distance from the two vEdge routers and because it may be situated behind a DMZ.

To allow vEdge routers at different overlay network sites on the private network to exchange data traffic directly using their private IP addresses, you configure their WAN interfaces to have one of the private colors, **metro-ethernet**, **mpls**, and **private1** through **private6**. Of these private colors, the WAN interfaces on the vEdge routers must be marked with the same color so that they can exchange data traffic.

Exchange Data Traffic within a Single Private WAN

To illustrate the exchange of data traffic across private WANs, let's look at a simple topology in which two vEdge routers are both connected to the same private WAN. The following figure shows that these two vEdge routers are connected to the same private MPLS network. The vEdge-1 router is located at Site 1, and vEdge-2 is at Site 2. Both routers are directly connected to PE routers in the carrier's MPLS cloud, and you want both routers to be able to communicate using their private IP addresses.



This topology requires a special configuration to allow traffic exchange using private IP addresses because:

- The vEdge routers are in different sites; that is, they are configured with different site IDs.
- The vEdge routers are directly connected to the PE routers in the carrier's MPLS cloud.

- The MPLS carrier does not advertise the link between the vEdge router and its PE router.

To be clear, if the situation were one of the following, no special configuration would be required:

- vEdge-1 and vEdge-2 are configured with the same site ID.
- vEdge-1 and vEdge-2 are in different sites, and the vEdge router connects to a CE router that, in turn, connects to the MPLS cloud.
- vEdge-1 and vEdge-2 are in different sites, the vEdge router connects to the PE router in the MPLS cloud, and the private network carrier advertises the link between the vEdge router and the PE router in the MPLS cloud.
- vEdge-1 and vEdge-2 are in different sites, and you want them to communicate using their public IP addresses.

In this topology, because the MPLS carrier does not advertise the link between the vEdge router and the PE router, you use a loopback interface on the each vEdge router to handle the data traffic instead of using the physical interface that connects to the WAN. Even though the loopback interface is a virtual interface, when you configure it on the vEdge router, it is treated like a physical interface: the loopback interface is a terminus for both a DTLS tunnel connection and an IPsec tunnel connection, and a TLOC is created for it.

This loopback interface acts as a transport interface, so you must configure it in VPN 0.

For the vEdge-1 and vEdge-2 routers to be able to communicate using their private IP addresses over the MPLS cloud, you set the color of their loopback interfaces to be the same and to one of private colors—**metro-ethernet**, **mpls**, and **private1** through **private6**.

Here is the configuration on vEdge-1:

```
vedge-1(config)# vpn 0
vedge-1(config-vpn-0)# interface loopback1
vedge-1(config-interface-loopback1)# ip address 172.16.255.25/32
vedge-1(config-interface-loopback1)# tunnel-interface
vedge-1(config-tunnel-interface)# color mpls
vedge-1(config-interface-tunnel-interface)# exit
vedge-1(config-tunnel-interface)# no shutdown
vedge-1(config-tunnel-interface)# commit and-quit
vedge-1# show running-config vpn 0
...
 interface loopback1
  ip-address 172.16.255.25/32
  tunnel-interface
   color mpls
  !
  no shutdown
 !
```

On vEdge-2, you configure a loopback interface with the same tunnel interface color that you used for vEdge-1:

```
vedge-2# show running-config vpn 0
vpn 0
 interface loopback2
  ip address 172. 17.255.26/32
  tunnel-interface
   color mpls
  no shutdown
 !
```

Use the **show interface** command to verify that the loopback interface is up and running. The output shows that the loopback interface is operating as a transport interface, so this is how you know that it is sending and receiving data traffic over the private network.

```
vedge-1# show interface

                                IF      IF                                                                  TCP
                                ADMIN   OPER    ENCAP                                       SPEED           MSS                     RX      TX
VPN  INTERFACE  IP ADDRESS      STATUS  STATUS  TYPE   PORT TYPE  MTU   HWADDR             MBPS    DUPLEX  ADJUST  UPTIME          PACKETS PACKETS
-------------------------------------------------------------------------------------------------------------------------------------------------
0    ge0/0      10.1.15.15/24   Up      Up      null   transport  1500  00:0c:29:7d:1e:fe  10      full    0       0:07:38:49      213199  243908
0    ge0/1      10.1.17.15/24   Up      Up      null   service    1500  00:0c:29:7d:1e:08  10      full    0       0:07:38:49      197     3
0    ge0/2      -               Down    Down    null   service    1500  00:0c:29:7d:1e:12  -       -       0       -               1       1
0    ge0/3      10.0.20.15/24   Up      Up      null   service    1500  00:0c:29:7d:1e:1c  10      full    0       0:07:38:49      221     27
0    ge0/6      57.0.1.15/24    Up      Up      null   service    1500  00:0c:29:7d:1e:3a  10      full    0       0:07:38:49      196     3
0    ge0/7      10.0.100.15/24  Up      Up      null   service    1500  00:0c:29:7d:1e:44  10      full    0       0:07:44:47      783     497
0    loopback1  172.16.255.25/32 Up     Up      null   transport  1500  00:00:00:00:00:00  10      full    0       0:00:00:20      0       0
0    system     172.16.255.15/32 Up     Up      null   loopback   1500  00:00:00:00:00:00  10      full    0       0:07:38:25      0       0
1    ge0/4      10.20.24.15/24  Up      Up      null   service    1500  00:0c:29:7d:1e:26  10      full    0       0:07:38:46      27594   27405
1    ge0/5      56.0.1.15/24    Up      Up      null   service    1500  00:0c:29:7d:1e:30  10      full    0       0:07:38:46      196     2
512  eth0       10.0.1.15/24    Up      Up      null   service    1500  00:50:56:00:01:05  1000    full    0       0:07:45:55      15053   10333
```

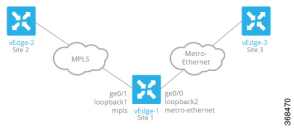
To allow vEdge routers at different overlay network sites on the private network to exchange data traffic directly, you use a loopback interface on the each vEdge router to handle the data traffic instead of using the physical interface that connects to the WAN. You associate the same tag, called a carrier tag, with each loopback interface so that all the routers learn that they are on the same private WAN. Because the loopback interfaces are advertised across the overlay network, the vEdge routers are able to learn reachability information, and they can exchange data traffic over the private network. To allow the data traffic to actually be transmitted out the WAN interface, you bind the loopback interface to a physical WAN interface, specifically to the interface that connects to the private network. Remember that this is the interface that the private network does not advertise. However, it is still capable of transmitting data traffic.

Share a Common Service across Different VPNs

When services such as firewalls or load balances are spread across multiple VPNs, you can create a policy that forces traffic from one VPN to use the services in another VPN. See the service control examples in Service Chaining Configuration Examples.

Use Case: Exchange Data Traffic between Two Private WANs

A variant of the topology illustrated above is the case in which a single vEdge router connects to two different private WANs, such as two different MPLS clouds provided by two different network carriers, or two different types of private WANs, as illustrated below. In this figure, the vEdge-1 router connects to one MPLS private WAN and one metro-Ethernet private WAN.



As in the previous example, you create loopback interfaces on the three routers. For vEdge-1, which connects to both of the private WANs, you create two loopback interfaces. For each one, you assign a color, as in the previous example. But you configure two more things: you assign a tag to identify the carrier, and you "bind" the loopback interface to the physical interface that connects to the private WAN. So, vEdge-1 has two loopback interfaces with these properties:

- Loopback1 has the color **mpls**, the carrier **carrier2**, and binds to physical interface ge0/1.
- Loopback 2 has the color **metro-ethernet** and the carrier **carrier1**, and binds to physical interface ge0/0.

The vEdge-2 router has a single loopback interface that connects to the MPLS private WAN. Its color is **mpls**, and its carrier is **carrier2**. Both these properties match those on the loopback1 interface on vEdge-1. However, because vEdge-2 connects to only one private WAN, there is no need to bind its loopback interface to a physical interface.

Finally, vEdge-3 has a single loopback interface with color **metro-ethernet** and carrier **carrier1**, matching the properties configured on the vEdge-1 loopback2 interface.

On vEdge-1, the configuration in VPN 0 looks like this:

```
vpn 0
 interface ge0/0
  ip address 10.1.15.15/24
  no shutdown
 !
 interface loopback2
  ip address 172.16.15.15/24
  tunnel-interface
   color   metro-ethernet
   carrier carrier1
   bind    ge0/0
  !
  no shutdown
 !

 interface ge0/1
  ip address 10.1.17.15/24
  no shutdown
 !
 interface loopback1
  ip address 172.16.17.15/24
  tunnel-interface
   color   mpls
   carrier carrier2
   bind    ge0/1
  !
  no shutdown
!
```

If you need to apply control policy to a particular private network, use the **match carrier** option when creating the control policy.

Segmentation CLI Reference

CLI commands for monitoring segmentation (VPNs) .

- **show bgp** commands
- **show interface** commands
- **show ospf** commands