

Configuring BGP/MPLS VPNs

3

This chapter describes how to configure BGP/MPLS VPNs on your ERX system. Before reading this chapter, you should be thoroughly familiar with both BGP and MPLS. For detailed information on those protocols, see *Chapter 1, Configuring BGP Routing* and *Chapter 2, Configuring MPLS*.

Topic	Page
Overview	3-1
References	3-19
Configuring BGP VPN Services	3-20
OSPF and BGP/MPLS VPNs	3-45
Monitoring BGP/MPLS VPNs	3-50

Overview

The BGP multiprotocol extensions (MP-BGP) enable BGP to support IPv4 services such as BGP Multicast and BGP/MPLS virtual private networks (VPNs). BGP/MPLS VPNs are sometimes known as RFC2547bis VPNs.

Address Families

The BGP multiprotocol extensions specify that BGP can exchange routing information for different types of *address families*. Our BGP implementation defines three different types of address families:

- Unicast IPv4 – When you enable BGP, the system employs unicast IPv4 addresses by default. You must specify an address family other than IPv4 for the new features.

- Multicast IPv4 – If you specify the multicast IPv4 address family, you can configure the system to exchange routes to multicast sources (as opposed to routes to unicast destinations). For a description of multicasting and information about BGP multicasting commands, see *Chapter 1, Configuring BGP Routing*.
- VPN-IPv4 – If you specify the VPN-IPv4 address family, you can configure the system to provide IPv4 VPN services via an MPLS backbone.

For information about specifying an address family, see *Configuring BGP VPN Services*, later in this chapter.

BGP/MPLS VPN Components

If you have specified the VPN-IPv4 address family, you can configure virtual private networks across an IP backbone. BGP carries routing information for the network and MPLS labels, whereas MPLS transports the data traffic. Figure 3-1 shows a typical scenario.

The service provider backbone comprises two types of routers:

- Provider edge routers (PEs)
- Provider core routers (Ps)

PE routers are situated at the edge of the service provider core and connect directly to customer sites. These routers must run BGP-4, including the BGP/MPLS VPN extensions. They must also be able to originate and terminate MPLS LSPs (see *Chapter 2, Configuring MPLS*, for more information).

Ps connect directly to PEs or other Ps and do not connect directly to customer sites. These routers must be able to switch MPLS LSPs—that is, they function as MPLS label-switching routers (LSRs) and might function as label edge routers (LERs). It is not necessary to run BGP-4 on the P routers to be able to exchange routing information for VPNs (although you may choose to run BGP-4 on the core routers for other reasons, such as exchanging routing information for the public Internet or implementing route reflectors). The P routes do not need to contain any information about customer sites.

PEs communicate with customer sites via a direct connection to a customer edge device (CE) that sits at the edge of the customer site. The CE can be a single host, a switch, or, most typically, a router. If the CE is a router, it is a routing peer of all directly connected PEs, but it is not a routing peer of CEs at any other site. The link between the CE and the PE can employ any type of encapsulation. It is not necessary to use MPLS. In

Figure 3-1, each PE connects to multiple CEs and at least one P. Although only one customer site is shown, each CE lies within a customer site.

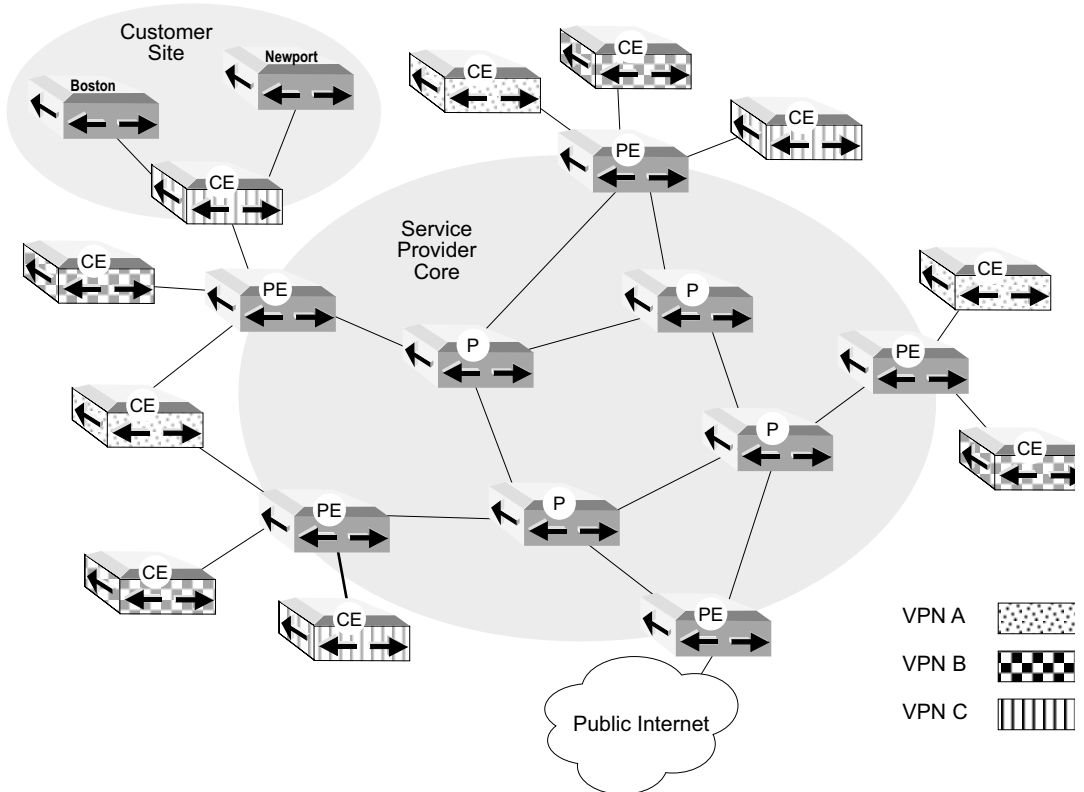


Figure 3-1 BGP/MPLS VPN scenario

A customer site is a network that can communicate with other networks in the same VPN. A customer site can belong to more than one VPN. Two sites can exchange IP packets with each other only if they have at least one VPN in common.

Each customer site that is connected to a particular PE is also associated with a VPN routing and forwarding instance (VRF). As shown in Figure 3-2, each VRF has its own forwarding table distinct from that of other VRFs and from the virtual router's global forwarding table.

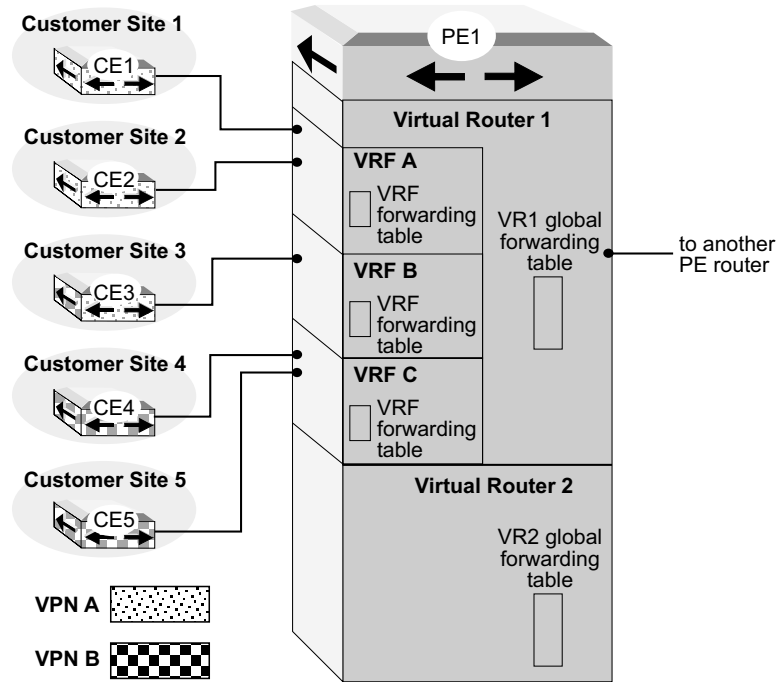


Figure 3-2 BGP/MPLS VPN components

A given VRF’s forwarding table includes only routes to sites that have at least one VPN in common with the site that is associated with the VRF. For example, in Figure 3-2, the forwarding table in VRF B stores routes only to sites that are members of at least one of the VPNs to which Customer Site 3 belongs.

VRFs exist within the context of a virtual router (VR). A given virtual router can have zero or more VRFs, in addition to its global routing table (which is not associated with any VPN, CE, or customer site). A system can support up to 1000 forwarding tables; that is, up to a total of 1000 VRs and VRFs.

You assign one or more (sub)interfaces to a given VRF. If multiple customer sites are members of the same set of VPNs, they can share a VRF—that is, it is not necessary to create a specific VRF for each customer site. In Figure 3-2, Customer Sites 1 and 2 share VRF A; both sites belong to the same set of VPNs. The system looks up a packet’s destination in the VRF associated with the interface on which the packet is received. The VRFs are populated by BGP as it learns routes from the VPN. If a customer site is a member of multiple VPNs, the routes learned from all those VPNs populate the VRF associated with the site.

VPN-IPv4 Addresses

Because each VPN has its own private address space, the same IP address might be used in several VPNs. To provide for more than one route to a given IPv4 address (each route unique to a single VPN), BGP/MPLS VPNs use route distinguishers (RDs) followed by an IPv4 address to create unique VPN-IPv4 addresses. A route can have only one RD.

The RD contains no routing information; it simply allows you to create unique VPN-IPv4 address prefixes. You can specify the RD in either of the following ways:

- An autonomous system (AS) number followed by a 32-bit assigned number. If the AS number is from the public address space, it must have been assigned to the service provider by the Internet Assigned Numbers Authority (IANA). The assigned number may be chosen by the service provider. Use of numbers from the private AS number space is strongly discouraged.
- An IP address followed by a 16-bit assigned number. If the IP address is from the public IP address space, it must have been assigned to the service provider by IANA. The assigned number may be chosen by the service provider. Use of numbers from the private IP address space is strongly discouraged.

One way to create unique VPN-IPv4 addresses is to assign a unique RD to each VRF in your network. However, the optimal strategy depends on the configuration of your network. For example, if each VRF always belongs to only one VPN, you could use a single RD for all VRFs that belong to a particular VPN.

Route Targets

A route-target extended community, or route target, is a type of BGP extended community that you use to define VPN membership. The route target appears in a field in the update messages associated with VPN-IPv4.

You create route-target import lists and route-target export lists for each VRF. The route targets that you place in a route target export list are attached to every route advertised to other PEs. When a PE receives a route from another PE, it compares the route targets attached to each route against the route-target import list defined for each of its VRFs. If any route target attached to a route matches the import list for a VRF, then the route is imported to that VRF. If no route target matches the import list, then the route is rejected for that VRF.

Depending on your network configuration, the import and export lists may be identical. Typically, you do the following:

- Allocate one route-target extended-community value per VPN.
- Configure the import list and the export list to include the same information: the set of VPNs comprising the sites associated with the VRF.

For more complicated scenarios—for example, hub-and-spoke VPNs—the route-target import list and the route-target export list might not be identical.

A route-target import list is applied before any inbound routing policy (route map) is applied. If an inbound route map contains a **set extcommunity** clause, the clause replaces all extended communities in the received route. BGP applies the default route-target export list associated with the VRF if the route does not have any route-target extended-community attributes after the inbound policy has been applied. On the other hand, the default export list is not applied if either a valid route-target export list is received or the inbound route map sets one or more route targets.

Distribution of Routes and Labels via BGP

The extensions to BGP include enhancements to update messages that enable them to carry the route distinguishers, route-target extended-community information, and MPLS labels required for BGP/MPLS VPNs.

Consider the simple example shown in Figure 3-3. The customer edge devices are connected with their associated provider edge routers via external BGP sessions (CE 1–PE 1 and CE 3 –PE 2). PE 1 and PE 2 are BGP peers via an internal BGP session across the service provider core in AS 777.

In this example, the PE routers run EBGP to the CE routers to do the following:

- Learn the prefixes of the networks in the local customer site.
- Advertise routes to networks and remote customer sites.

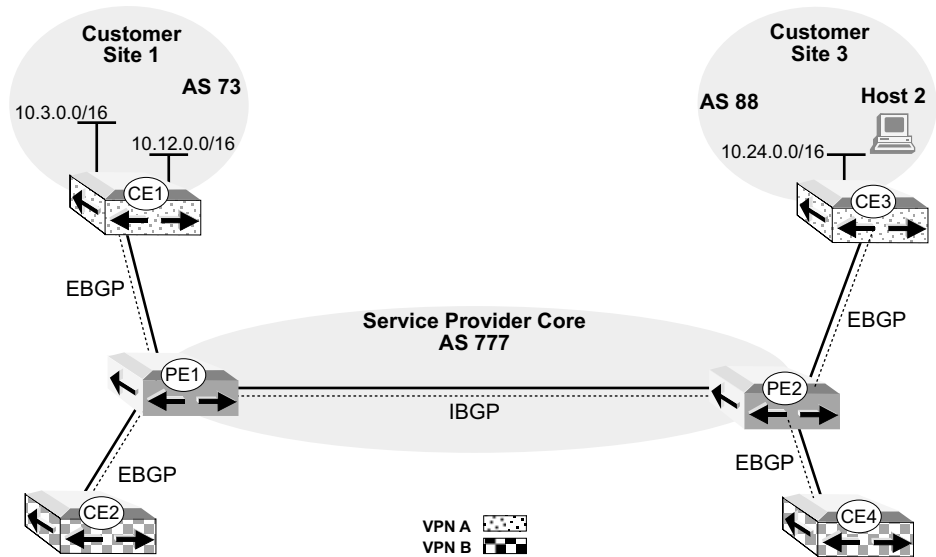


Figure 3-3 Route and label distribution

Running EBGP between the PEs and the CEs is not the only way to accomplish those tasks. Alternatively, you could do either of the following:

- Run an IGP (such as IS-IS, OSPF, or RIP) between the CE and the PE.
- Configure static routes on the CE and PE (on the CE this would typically be a default route).

Also note in this example that the two customer sites use different AS numbers. This simplifies configuration, but it is also possible to use the same AS numbers.

Customer site 1 has two networks that need to be reachable from customer site 3—10.3.0.0/16 and 10.12.0.0/16—and uses BGP to announce these prefixes to PE 1. CE 1 uses a standard BGP update message as shown in Figure 3-4 to carry this and additional information. CE 1 is withdrawing prefix 10.1.0.0/16. CE 1 specifies its own address as the next hop; 10.4.1.1 is from the private address space of VPN A.

PE 1 passes the advertisement along the backbone via an IBGP session, but uses MP-BGP rather than standard BGP-4. Consequently, PE 1 uses an extended BGP update message, which is quite different in format from the standard message, as shown in Figure 3-4.

The extended update uses different attributes for some of the advertised information. For example it carries the advertised prefixes in the

MP-Reach-NLRI attribute instead of the NLRI (network layer reachability information) attribute. Similarly, it uses the MP-Unreach-NLRI attribute for withdrawn routes rather than the withdrawn-routes attribute.

PE 1 advertises the customer site addresses by prepending information to the addresses as advertised by CE 1, thus creating *labeled VPN-IPv4 prefixes*. The prepended information consists of a route distinguisher and an MPLS label.

Because the CE uses IPv4 addresses from the VPN's private address space, these addresses could be duplicated in other VPNs to which PE 1 is attached. PE 1 associates a route distinguisher with each IPv4 address to create a globally unique address. In this example, the RD consists of the AS that PE 1 belongs to and a number that PE 1 assigns. The RD is prepended immediately before the IPv4 address.

PEs assign MPLS labels to each VRF. In this example, the label for the VRF associated with customer site 1 is 16. The MPLS label is prepended immediately before the route distinguisher.

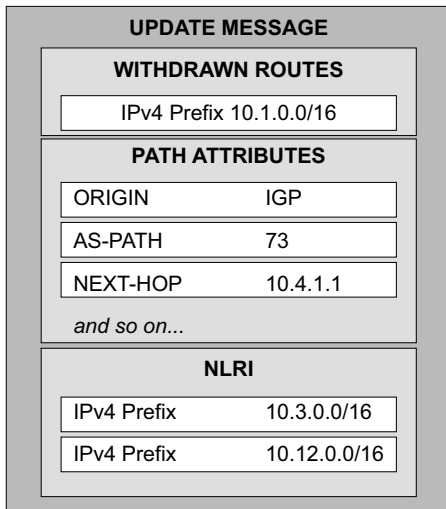


Note: *The explicit null label is prepended only to routes that are being withdrawn in the MP-REACH-NLRI attribute.*

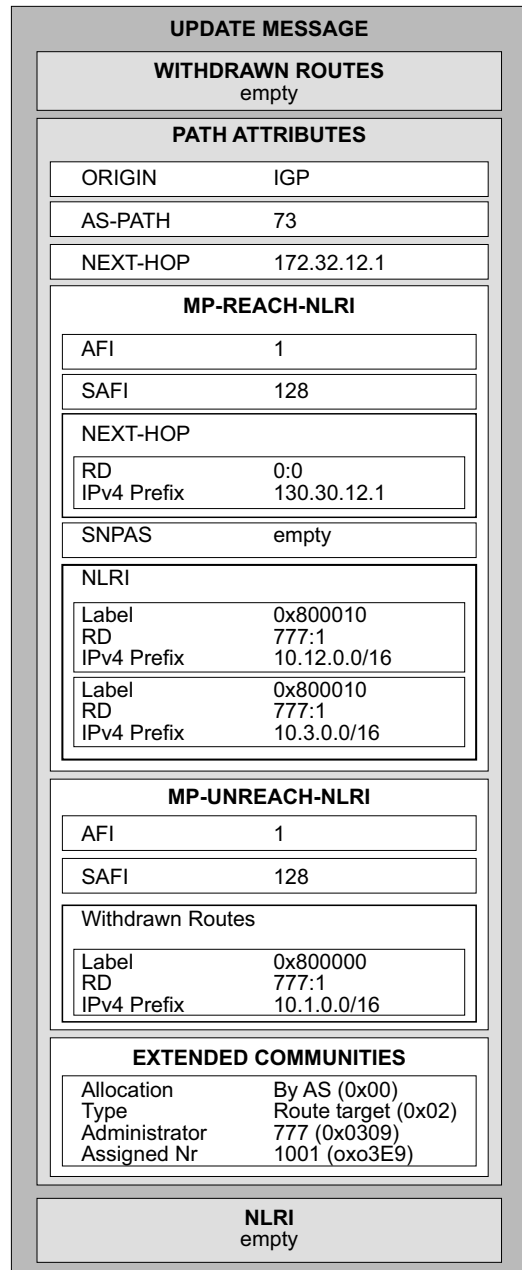
Some non-ERX implementations allocate a separate label for each prefix. By default, the ERX system generates one label for all BGP routes advertised by the VRF, thus reducing the number of stacked labels to be managed. The **ip mpls forwarding-mode label-switched** command enables you to have the system generate a label for each different FEC pointed to by a BGP route in a given VRF. However, some routes always receive a per-VRF label; see *Creating Labels Per FEC* later in this chapter for more information.

Using the **next-hop-self** option on PE 1 causes PE 1 to set the next-hop attribute to its own address, 172.32.12.1. This is necessary because the next hop provided by CE 1 is from VPN A's private address space and has no meaning in the service provider core. In addition, PE 2 must have PE 1's address so that it can establish an LSP back to PE 1. The next-hop address must also be carried in the MP-Reach-NLRI attribute, according to MP-BGP.

The extended update also has the extended-communities attribute, which identifies the VPN to which the routes are advertised. In this example, the route target is 777:1001, identifying VPN A.



Standard update message



Extended update message

Figure 3-4 Standard and extended BGP update messages

VPN Interface Allocation and Scaling

Some non-ERX systems assign a label for each prefix that a PE advertises over the BGP/MPLS VPN. By default, the system receiving the prefix/label pairs always allocates a VPN interface only for each next-hop PE, rather than for each received stacked label, enabling the system to support a very large number of egress FECs. If the ERX system instead allocated a VPN interface for each received stacked label, scalability would suffer if the non-ERX PE advertised a very large number of prefix/label pairs.

All labels received from a given next hop are stored in that interface, called a variable interface. Statistics are collected for the variable interface as a whole; label-specific statistics are not available. If you want to see statistics on a per-label basis, you can use the **ip mpls vpn-interface per-label** command to create a VPN interface for each received stacked label. However, operating in this per-label mode limits the number of egress FECs supported to the order of thousands.



Note: If you change between VPN interface allocation modes (from allocating a VPN interface per label to allocating a VPN interface per next hop or vice-versa), the next hops for the BGP/MPLS VPN routes become unreachable and will slowly be resolved again. This happens because the stacked tunnels to which the next hops are resolved are removed and recreated in the new mode.

Transporting Packets Across an IP Backbone via MPLS

As described in the previous section, PE 1 and PE 2 exchange routing information, including MPLS labels for their customer sites, via a BGP session established between them across the service provider core.



Note: This section will be more easily understood if you have an understanding of MPLS. See Chapter 2, *Configuring MPLS*, for a detailed explanation of MPLS.

Labels are employed in both the BGP control plane and the MPLS data plane. In the control plane, BGP advertises a route with an in label; this in label is also the label that is expected when MPLS traffic is received. BGP receives routes with an associated out label; the out label is the label sent with MPLS traffic.

Consider the network shown in Figure 3-5. If you display the in label on PE1, you see that MP-BGP advertises a labeled VPN-IPv4 prefix of 10.12.0.0/16 with an in label of 24 (and an RD of 777:1, as shown in illustration).

```
host1:pe1#show ip bgp vpn all field in-label
Prefix          In-label
10.12.0.0/16    24
10.24.0.0/16    none
```

If you display the in label on PE2, you see that MP-BGP advertises a labeled VPN-IPv4 prefix of 10.24.0.0/16 with an in label of 16 (and an RD of 777:5, as shown in illustration).

```

host2:pe2#show ip bgp vpn all field in-label
Prefix          In-label
10.12.0.0/16   none
10.24.0.0/16   16
    
```

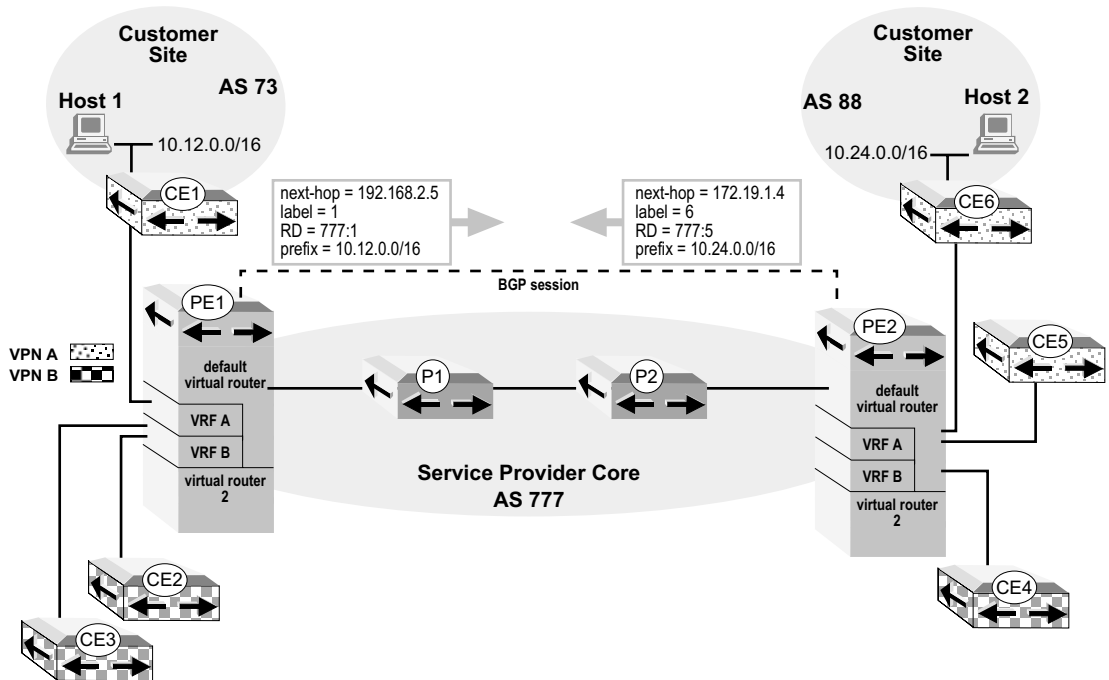


Figure 3-5 BGP/MPLS VPN route exchange

Back on PE1, you see that MP-BGP receives a labeled VPN-IPv4 prefix of 10.24.0.0/16 with an out label of 16. This is the label that MP-BGP on PE2 advertised with the prefix. In the data plane, MPLS traffic is sent by PE1 to PE2 with this label.

```

host1:pe1#show ip bgp vpn all field out-label
Prefix          Out-label
10.12.0.0/16   none
10.24.0.0/16   16
    
```

On PE2, you see that MP-BGP receives a labeled VPN-IPv4 prefix of 10.12.0.0/16 with an out label of 24. This is the label that MP-BGP on PE1 advertised with the prefix. In the data plane, MPLS traffic is sent by PE2 to PE1 with this label.

```
host2:pe2#show ip bgp vpn all field out-label
Prefix          Out-label
10.12.0.0/16    24
10.24.0.0/16    none
```

The data packets are transported within a VPN across the service provider core via MPLS. This transport process requires two layers of MPLS labels, stacked one upon the other.

The inner labels are those assigned by each PE for each VRF. When an MPLS packet arrives at the egress PE, that egress PE uses the inner label to determine which VRF the packet is destined for. In the default, per-VRF label allocation mode (see *Creating Labels Per FEC* later in this chapter), the egress PE does an IP lookup in the IP forwarding table of that VRF using the IP destination address in the IP packet that is encapsulated in the MPLS packet. The egress PE then forwards the IP packet (without the MPLS header) to the appropriate customer site. The inner labels themselves are communicated between PEs in the MP-BGP extended update messages as described in the previous section.

MPLS uses the outer labels to forward data packets from the ingress PE through a succession of P routers across the core. This succession of P routers constitutes a label-switched path (LSP), also referred to as an MPLS tunnel. The labels are assigned to links in the path.

At each P router, MPLS pops the outer label from a data packet. The label is an index into the P router's forwarding table, from which it determines both the next hop along the LSP and another label. The router pushes the label on to the label stack and forwards the packet to the next P router. The combination of popping one label and pushing another is known as a label swap. At the egress PE, MPLS pops the outer label, then the inner label. The inner label determines the CE to which the packet is sent. The P routers never look at the inner MPLS label or the destination IP address encapsulated in the MPLS packet.

In many cases, the PEs will be fully meshed via LSPs. You can use tunnel profiles to simplify the LSP configuration process. See *Chapter 2, Configuring MPLS*, for procedures to configure an LSP.

Each LSP is unidirectional for data traffic, so you must establish LSPs in both directions for two-way data transport. Figure 3-6 shows that two LSPs have been created between PE 1 and PE 2. PE 1 and PE 2 have an MP-BGP session as shown previously in Figure 3-5.

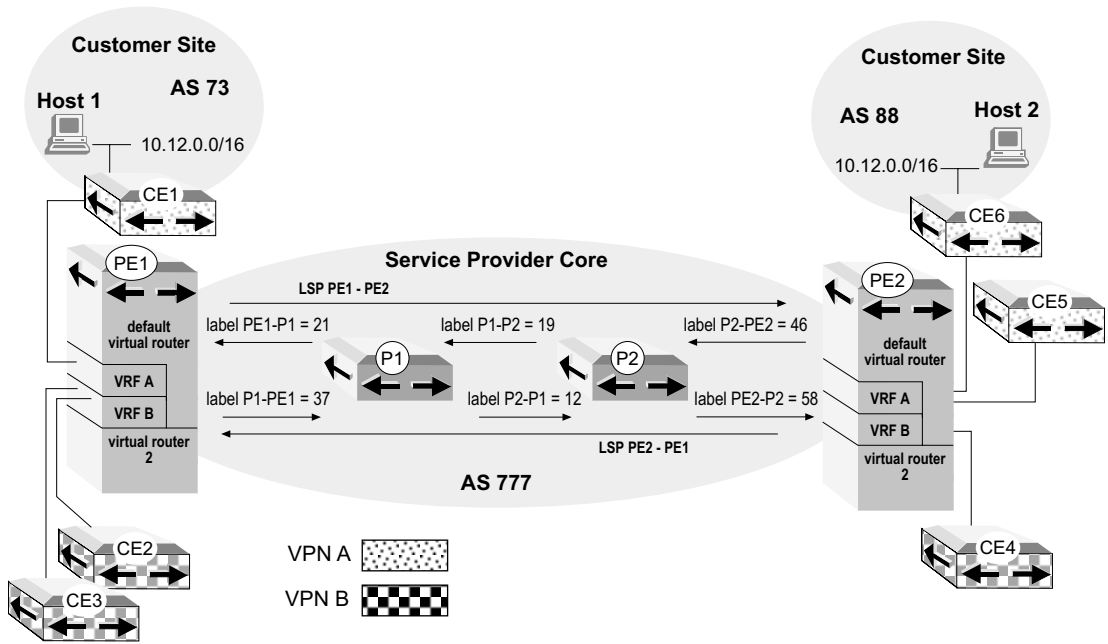


Figure 3-6 LSP creation for BGP/MPLS VPN

The PE 1–PE 2 LSP carries traffic only from PE 1 to PE 2, using label 21 for the PE 1 to P 1 link, label 19 for the P 1 to P 2 link, and label 46 for the P 2 to PE 2 link. PE 1 can forward data packets along the LSP to PE 2 and its customer sites.

Similarly, the PE 2–PE 1 LSP carries traffic only from PE 2 to PE 1, using label 58 for the PE 2 to P 2 link, label 12 for the P 2 to P 1 link, and label 37 for the P 1 to PE 1 link. PE 2 can forward data packets along the LSP to PE 1 and its customer sites.

Example: The process of data transport is shown in Figure 3-7. PE 1 has already received announcements from PE 2; an LSP has been established between PE 1 and PE 2.
Data Transport

Host 1 constructs an IP packet with the address of Host 2 as the final destination, and sends the packet to router CE 1. CE 1 encapsulates the packet appropriately and forwards it to PE 1.

PE 1 receives the packet from CE 1. Based on the interface the packet came in on, PE 1 determines that it must use the forwarding table for VRF A to route the packet. PE 1 looks up the destination address of Host 2 in the forwarding table of VRF A and finds the following instructions:

- Push label 16; that is, prepend it to the data packet. This is the innermost label that identifies the VRF on PE 2, where the final destination and interface lookup takes place. Label 16 was previously allocated by PE 2 and communicated to PE 1 via MP-BGP. VRF A shows this label as part of the NLRI for destination address Host 2.
- Push label 21 and forward the MPLS-encapsulated data packet to router P 1. Label 21 is prepended to label 16; the labels are *stacked*. Label 21 becomes the outermost label and is assigned to the first segment—PE 1–P 1—in the label-switched path from PE 1 to PE 2. The LSP was previously configured.

P 1 receives the data packet from PE 1 and pops label 21. P 1 looks up label 21 in its forwarding table and determines it must push label 19 on the stack, and forwards the data packet to P 2.

P 2 receives the data packet from P 1 and pops label 19. P 2 looks up label 19 in its forwarding table and determines it must push label 46 on the stack, and forwards the data packet to PE 2.

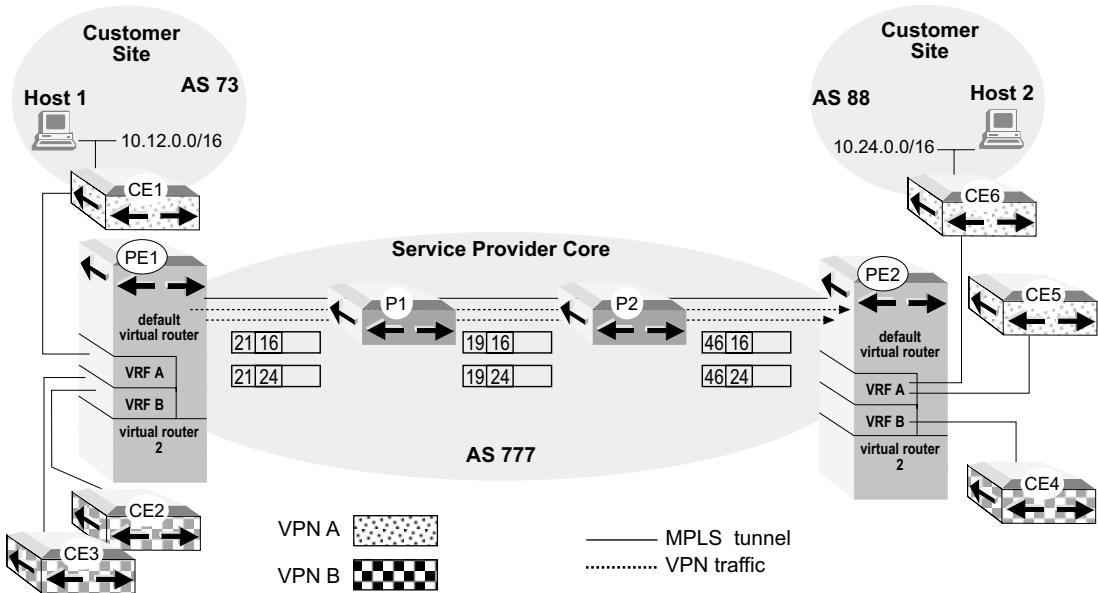


Figure 3-7 Traffic across the MPLS backbone of a BGP/MPLS VPN

PE 2 receives the data packet from P 2, and looks up label 46. PE 2 determines it is the egress router of the LSP and must pop label 46. Then it proceeds to look up the next label, label 16, and determines that the packet goes to VRF A. Then the IP address is looked up in VRF A to

determine the destination and outgoing interface for the packet. PE 2 forwards the packet to CE 6.

CE 6 receives the IP packet from PE 2 and looks up the destination address Host 2. Subsequent forwarding to Host 2 occurs via the IGP in the customer site.

The network structure shown in Figure 3-7 consists of two VPNs, A and B. VPN A comprises CE 1, CE 5, and CE 6. VPN B comprises CE 2, CE 3, and CE 4. CE 1 has data traffic destined for both CE 5 and CE 6. Because both of these destination sites are within the same VPN, PE 1 uses the same forwarding table, in VRF A, to do the lookups and MPLS encapsulation. The innermost label determines the destination VRF and is the same for all packets in that VPN, even if they are destined for different CEs. CE 2 and CE 3 have traffic destined for CE 4. Because these all are in VPN B, PE 1 uses a different forwarding table, in VRF B, for looking up destinations for traffic originating with these sites. However, both VPNs use the same LSP, because both VPNs use the same ingress (PE 1) to and the same egress (PE 2) from the service provider core. Remember that the illustrated LSP carries data traffic only from PE 1 to PE 2. Traffic from PE 2 to PE 1 requires a different LSP.

Load Balancing

The ERX system supports load balancing across multiple LDP base LSPs. Formerly, a given PE supported only a single next hop for all prefixes and could not take advantage of multiple available next hops. Now the system uses a hash-based algorithm to select from among all available equal-cost next hops for each prefix.

Traffic to a given prefix is transmitted across only a single LSP. For contiguous destination prefixes, the ERX system load balances the total traffic equally across all available LDP base LSPs in round-robin fashion. For noncontiguous destination prefixes, the traffic is spread less equally. If an LSP becomes unavailable, the traffic is balanced across the remaining available LSPs.

Providing VPN Services Across Multiple Autonomous Systems

Inter-AS services, sometimes known as interprovider services, support VPNs that cross AS boundaries. VPNs might need to cross AS boundaries because of a customer deployment that involves geographically separated ASs. The VPN sites can be provided by the same service provider or by different service providers as part of a joint VPN service offering. Inter-AS services are also useful to service providers that use confederations of sub-ASs to reduce the IBGP mesh inside the AS.

You can support these inter-AS services in two ways. In the first method (Figure 3-8), you create a VRF for each VPN on each AS border router (ASBR). Within each AS, routes are announced via internal MP-BGP and the data packets are forwarded across an MPLS tunnel. You create a logical connection such as an ATM VC between each pair of VRFs (on separate ASBRs); these logical connections can share the same physical connection. The following factors limit the scalability of this method:

- All inter-AS VPN routes (potentially a very large number) must be stored in the BGP RIBs and IP routing tables on the ASBRs.

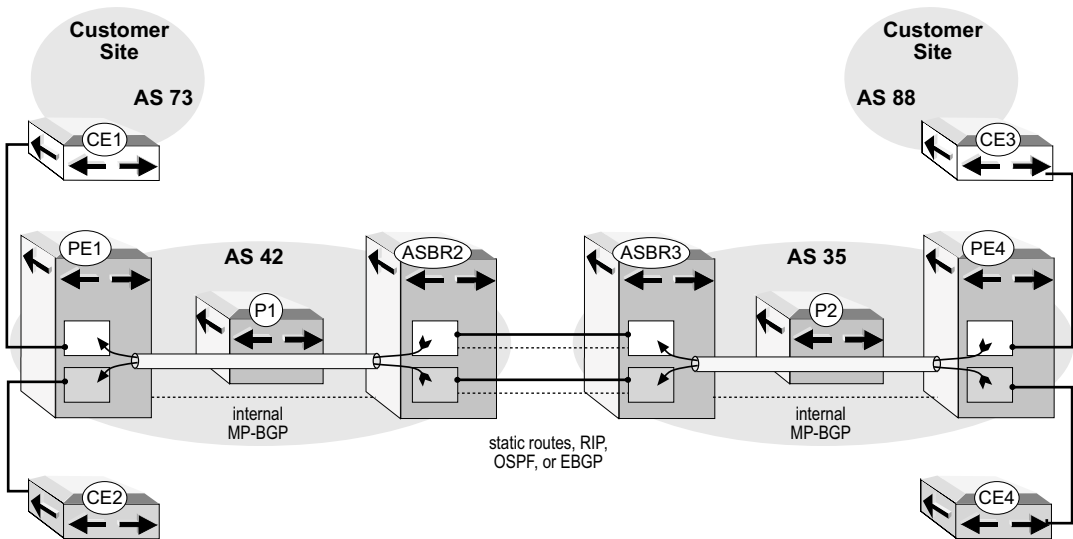


Figure 3-8 Inter-AS topology with VRFs on each ASBR

MPLS tunnels are unidirectional; Figure 3-8 shows only the tunnels established to carry traffic from ASBR 2 to PE 1 and from PE 4 to ASBR 3. Note that ASBR 2 and ASBR 3 are both also PEs. In that sense, ASBR 2 treats ASBR 3 as a CE, and ASBR 3 treats ASBR 2 as a CE.

The second method is the one we recommend and is often referred to as 2547 option B, after IETF draft RFC (BGP/MPLS VPNs – draft-ietf-ppvpn-rfc2547bis-01.txt). This method uses BGP to signal VPN labels between the ASBRs (Figure 3-9). The base MPLS tunnels are local to each AS. Stacked tunnels run from end to end between PEs on the different ASs. This method provides greater scalability, because only the BGP RIBs store all the inter-AS VPN routes.

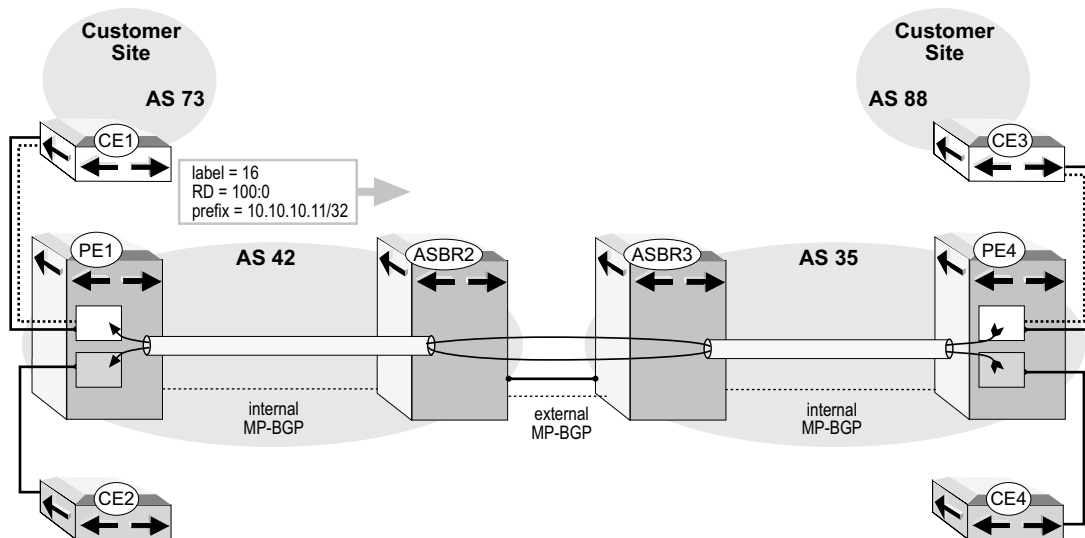


Figure 3-9 Inter-AS topology with end-to-end stacked MPLS tunnels

PE 1 assigns labels for routes to the customer sites, and distributes both the label assignments and the VPN-IPv4 routes throughout AS 42 in extended BGP update messages via internal MP-BGP. ASBR 2 then distributes the routes to ASBR 3 via external MP-BGP; ASBR 2 specifies itself as the next-hop address and assigns a new label to the route so that ASBR 3 can properly direct traffic. ASBR 3 propagates the routes via internal MP-BGP throughout AS 35, including to PE 4.

Example You can use the **show ip bgp vpn all field in-label** and **show ip bgp vpn all field out-label** commands in the context of each VPN element to display the in label and out label associated with the route at that point. Suppose that CE 1 advertises a route to prefix 10.10.10.11/32 to its external BGP peer PE 1 (10.2.2.2) in VRF A. PE 1 associates the label 16 with this route; an extended update message sent to intranl MP-BGP peer ASBR 2 carries this information as a labeled VPN-IPv4 prefix (label 16, RD 100:0, IPv4 prefix 10.10.10.11/32).

```
host1:pe1#show ip bgp vpn all field in-label
Prefix          In-label
10.10.10.11/32  16
```

On PE 1, there is no out label associated with the IPv4 prefix 10.10.10.11/32.

```
host1:pe1#show ip bgp vpn all field out-label
Prefix          Out-label
10.10.10.11/32  none
```

ASBR 2 receives the labeled VPN-IPv4 prefix and generates a new label, 44, to associate with this VPN-IPv4 prefix instead of label 16 when it sends the prefix to ASBR 3.

```
host1:asbr2#show ip bgp vpn all field out-label
Prefix                Out-label
10.10.10.11/32        16

host1:asbr2#show ip bgp vpn all field in-label
Prefix                In-label
10.10.10.11/32        44
```

Notice that ASBR2 receives MPLS frames with label 44 (the in label) and sends MPLS frames with label 16 (the out label).

The inter-AS next hop shows label 44 as the label advertised to inter-AS peer ASBR 3. Label 44 was generated for the indirect next hop PE/label pair, 10.2.2.2 (PE 1) and 16.

```
host1:asbr2#show ip bgp vpn all next-hops
Indirect next-hop 10.2.2.2
  type MPLS INTER-AS advertising case label 44
  Reference count is 1
```

ASBR 3 in turn generates a new label, 50, to advertise with the VPN-IPv4 prefix to its internal MP-BGP peer inside its autonomous system, AS 35.

```
host1:asbr3#show ip bgp vpn all field out-label
Prefix                Out-label
10.10.10.11/32        44

host1:asbr3#show ip bgp vpn all field in-label
Prefix                In-label
10.10.10.11/32        50

host1:asbr3#show ip bgp vpn all next-hops
Indirect next-hop 10.21.21.1
  type MPLS INTER-AS receiving case label 50
  Using EBGp peer incoming interface ATM6/1.21
  Reference count is 1
```

In turn, ASBR3 receives MPLS frames with label 50 (the in label) and sends MPLS frames with label 44 (the out label).

PE 4 receives the VPN-IPv4 prefix with label 50:

```
host1:pe4#show ip bgp vpn all field out-label
Prefix                Out-label
10.10.10.11/32        50
```

On PE 4, there is no in label associated with the IPv4 prefix 10.10.10.11/32.

```

host1:pe4#show ip bgp vpn all field in-label
Prefix                In-label
10.10.10.11/32        none

```

The labels that are generated to be sent to the inter-AS BGP peers are generated for each next-hop PE/received label tuple. Scaling is improved when all routes advertised from a given VRF have the same label; this is the default ERX system behavior. You can disable this behavior by issuing the **ip mpls forwarding-mode label-switched** command for the VRF.



Note: *The inter-AS BGP peers must be connected via a common interface—so that the number of next hops between them is zero; that is, they must be nonmultihop peers. EBGP multihop is not supported for inter-AS BGP VPNs. Further, the inter-as BGP peers cannot be members of a peer group.*

References

For more information about the BGP/MPLS VPNs, consult the following resources:

- BGP/MPLS VPNs – draft-ietf-ppvpn-rfc2547bis-03.txt (April 2003 expiration)
- *ERX Release Notes, Appendix A, System Maximums* – refer to the Release Notes corresponding to your software release for information on maximum values.
- RFC 2547 – BGP/MPLS VPNs (March 1999)
- RFC 2858 – Multiprotocol Extensions for BGP-4 (June 2000)
- RFC 3107 – Carrying Label Information in BGP-4 (May 2001)

For more information about BGP and MPLS see the *Reference* sections in *Chapter 1, Configuring BGP Routing* and *Chapter 2, Configuring MPLS*.

Configuring BGP VPN Services

To configure a system to provide BGP VPN services, you must perform some tasks once per PE and some tasks for each VRF on the PE.

VRF Configuration Tasks

To configure a VRF to provide BGP VPN services:

- 1 Create the VRF.

```

host1(config)#virtual-router vr1

```

```
host1:vr1(config)#ip vrf vrfA
```

- 2 Assign a route distinguisher to the VRF.

```
host1:vr1(config-vrf)#rd 100:100
```

- 3 Set the route-target import and route-target export lists for the VRF.

```
host1:vr1(config-vrf)#route-target import 100:1  
host1:vr1(config-vrf)#route-target export 100:1
```

- 4 (Optional) Set import and export maps for the VRF.

```
host1:vr1(config-vrf)#import map Another-route-map  
host1:vr1(config-vrf)#export map A-route-map  
host1:vr1(config-vrf)#exit
```

- 5 Assign interfaces for PE-to-CE links to the VRF from inside or outside the VRF context:

```
host1:vr1(config)#interface gigabitEthernet 1/0  
host1:vr1(config-if)#ip vrf forwarding vrfA  
host1:vr1(config-if)#ip address 10.16.2.77 255.255.255.0  
host1:vr1(config-if)#exit
```

or

```
host1:vr1(config)#virtual-router :vrfA  
host1:vr1:vrfA(config)#interface gigabitEthernet 1/0
```

- 6 Use either of the following methods to establish how the VRF learns routes to customer sites:

- Create static routes to the customer site in the VRF by one of the following methods:

```
host1(config)#virtual-router vr1  
host1:vr1(config)#ip vrf vpnA  
host1:vr1(config-vrf)#ip route vrf vrfA 10.3.0.0 255.255.0.0  
10.1.1.1  
host1:vr1(config-vrf)#ip route vrf vrfA 10.12.0.0  
255.255.0.0 10.1.1.1
```

or

```
host1(config)#virtual-router vr1:vrfA  
host1:vr1:vrfA(config)#ip route 10.3.0.0 255.255.0.0  
10.1.1.1  
host1:vr1:vrfA(config)#ip route 10.12.0.0 255.255.0.0  
10.1.1.1
```

- Configure an IGP on the VRF to learn routes from the CE.

See *Configuring IGPs on the VRF* later in this chapter for examples.

- Configure a PE-to-CE EBGp session.

See *Configuring PE-to-CE BGP Sessions* later in this chapter for information on configuring EBGp.

- 7 (Optional) Configure the system to generate a label for each different FEC pointed to by a BGP route in the VPN.

```
host1:vr1(config-vrf)#ip mpls forwarding-mode label-switched
```

- 8 (Optional) Configure the system to create a VPN interface for each received stacked label, enabling collection of statistics on a per-label basis.

```
host1:vr1(config-vrf)#ip mpls vpn-interface per-label
```

PE Configuration Tasks

To configure a PE to provide BGP VPN services:

- 1 Configure PE-to-PE LSPs.

See *Chapter 2, Configuring MPLS*, for information on configuring LSPs.

- 2 Enable BGP routing.

```
host1:vr1(config)#router bgp 100
```

- 3 (Optional) Disable automatic route-target filtering.

```
host1:vr1(config-router)#no bgp default route-target filter
```

- 4 Configure PE-to-PE BGP sessions.

- a Create the PE-to-PE session.

```
host1:vr1(config)#router bgp 100
```

```
host1:vr1(config-router)#neighbor 192.168.1.158 remote-as 100
```

- b Create the VPN-IPv4 address family.

```
host1:vr1(config-router)#address-family vpnv4
```

- c Activate the PE-to-PE session in the VPN-IPV4 address family.

```
host1:vr1(config-router-af)#neighbor 192.168.1.158 activate
```

```
host1:vr1(config-router-af)#exit-address-family
```

- 5 Configure PE-to-CE BGP sessions.

- a Enable and configure BGP:

```
host1:vr1(config)#router bgp 100
```

See *Chapter 1, Configuring BGP Routing*, for more information on configuring BGP.

- b Specify the IPv4 unicast address family for each VRF:

```
host1:vr1(config-router)#address-family ipv4 unicast vrf
vrfA
```

- c Configure the method of route advertisement by doing one of the following:

- Use **neighbor** commands to specify peers to which BGP advertises the routes:

```
host1:vr1(config-router)#neighbor 10.12.13.0 remote-as 200
```

- Use **network** commands or the **redistribute static** command to make BGP advertise static routes to customers.

```
host1:vr1(config-router)#network 10.3.0.0 mask 255.255.0.0
host1:vr1(config-router)#redistribute static
```

- Use **redistribute** commands to make BGP advertise IGP routes to customers.

```
host1:vr1(config-router)#redistribute ospf
```

- 6 (Optional) Configure an AS override.

See *Using a Single ASN for all CE Sites* later in this chapter for examples.

- 7 (Optional) Force the BGP speaker to accept routes that have the speaker's AS number in its AS path.

```
host1:vr1(config-router)#bgp enforce-first-as
```

Creating a VRF

Access the desired virtual router context; then create the VRF(s) for that VR.

```
host1(config)#virtual-router vr1
host1:vr1(config)#ip vrf vrfA
```

ip vrf

- Use to create a VRF or access VRF Configuration mode to configure a VRF.
- You must specify a route distinguisher after you create a VRF. Otherwise, the VRF will not operate.

- Example

```
host1:vr1(config)#ip vrf vrfA
```

- Use the **no** version to remove a VRF.

Specifying a Route Distinguisher

The route distinguisher enables you to establish unique VPN-IPv4 addresses to accommodate the possibility that more than one VPN might use the same IP address from their private address spaces.

rd

- Use to specify a route distinguisher to a VRF.
- You can specify either an AS number or an IP address as the first part of the route distinguisher. Specify some unique integer as the second part.
- You must specify a route distinguisher for a VRF. Otherwise, the VRF will not operate.
- Once you have configured the route distinguisher, you can change it only by removing and recreating the VRF.

- Example

```
host1:vrl(config-vrf)#rd 100:100
```

- There is no **no** version.

Defining Route Targets for VRFs

BGP uses an extended-community attribute, the *route target*, to filter appropriate VPN routes into the correct VRFs. You configure the *export list* on the VRF to specify export route targets. When BGP advertises a route from this VRF's forwarding table, it associates the list of export route targets with the route and includes this attribute in the update message that advertises the route.

You also configure a route-target *import list* on each VRF to specify import route targets. When a PE receives a route, BGP compares the route target list associated with the route (and carried in the update message) with the import list associated with each VRF configured in the PE.

For VPN-IPv4 routes received from another PE, if *any* route target in the export list matches a route target in a VRF's import list, then the route is installed in that VRF's forwarding table.

Although it is possible to create very sophisticated route distribution schemes, the most common configuration is to do the following:

- Allocate one route-target extended-community value per VPN.
- Define the route-target import list and a route-target export list to include only the route-target extended-community values for the VPN(s) to which the VRF belongs:

```
host1:vrl(config-vrf)#route-target export 777:100  
host1:vrl(config-vrf)#route-target import 777:100
```

If the import and export lists are identical, you can use the **both** keyword to define the lists simultaneously:

```
host1:vr1(config-vrf)#route-target both 777:105
```

A route-target export list can be modified on the sending PE by an export map or outbound routing policy. It can be modified on the receiving PE by an import map or inbound routing policy.

route-target

- Use to create—or add to—lists of VPN extended communities for a VRF that determine whether a route is imported into a VRF.
- An export list defines a route-target extended community; routes having any route target in their export list that matches a route target in a VRF's import list are installed in the VRF's forwarding table.
- An import list defines a route-target extended community; only routes that have at least one matching route target in their associated export list can be installed into the VRF's forwarding table.
- If the import and export lists are identical, use the **both** keyword to define both lists simultaneously.
- You can add only one route target to a list at a time.
- Example

```
host1:vr1(config-vrf)#route-target export 100:1  
host1:vr1(config-vrf)#route-target import 100:1
```

- Use the **no** version to remove a route target from the import list, the export list, or both lists.

Example: Fully Meshed VPNs

In a fully meshed VPN, each site in the VPN can reach every other site in the VPN. Figure 3-10 illustrates a situation with two fully meshed VPNs, VPN A and VPN B. VPN A includes Customer Sites 1, 3, and 5 through VRFs A, C, and E. VPN B includes Customer Sites 2, 4, and 6 through VRFs B, D, and F.

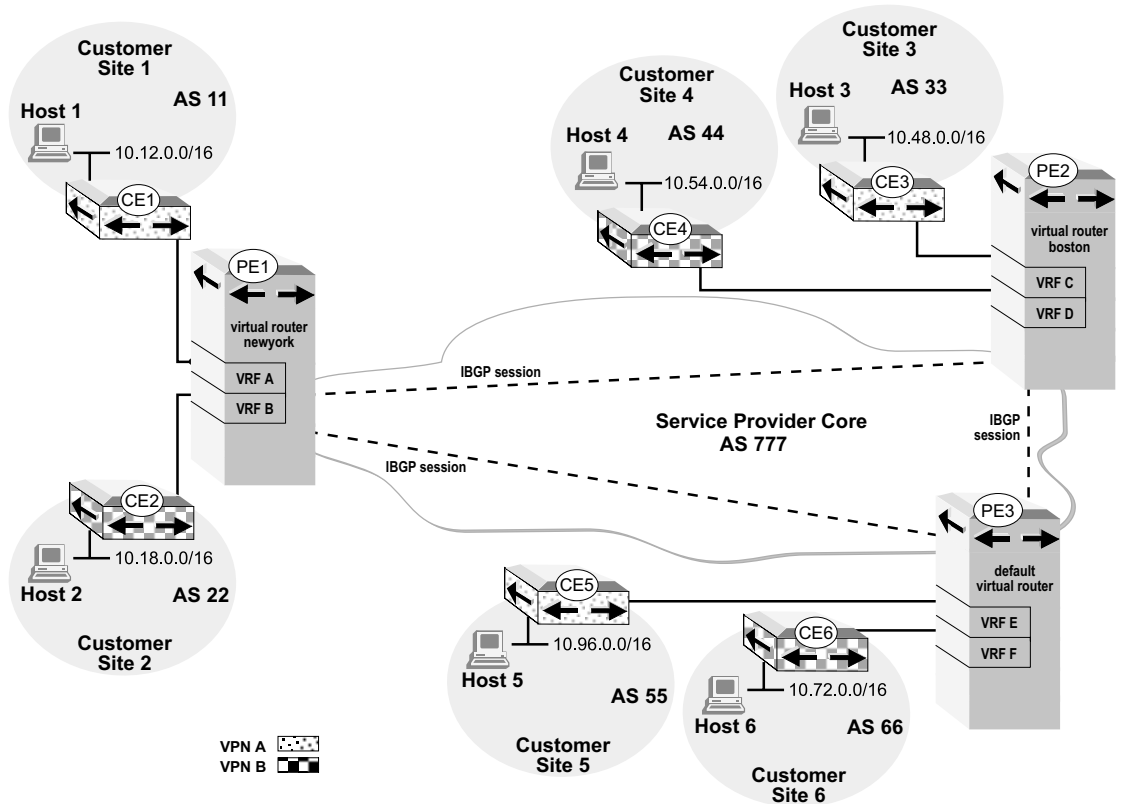


Figure 3-10 Fully meshed VPNs

BGP sessions exist between PE 1 and PE 2, PE 2 and PE 3, and PE 3 and PE 1. The MPLS paths through the service provider core are omitted for clarity.

To configure route targets for this fully meshed scenario, you specify the same route target for the import list and export list on all VRFs in VPN A. The VRFs in VPN B use a different route target, but it is the same for the import list and export list for all.

Route-target configuration on PE 1:

```

host1(config)#virtual-router newyork
host1:newyork(config)#ip vrf vrfA
host1:newyork(config-vrf)#route-target both 777:1
host1:newyork(config-vrf)#exit
host1:newyork(config)#ip vrf vrfB
host1:newyork(config-vrf)#route-target both 777:2
    
```

Route-target configuration on PE 2:

```
host2(config)#virtual-router boston
host2:boston(config)#ip vrf vrfC
host2:boston(config-vrf)#route-target both 777:1
host2:boston(config-vrf)#exit
host2:boston(config)#ip vrf vrfD
host2:boston(config-vrf)#route-target both 777:2
```

Route-target configuration on PE 3:

```
host3(config)#ip vrf vrfE
host3(config-vrf)#route-target both 777:1
host3(config-vrf)#exit
host3(config)#ip vrf vrfF
host3(config-vrf)#route-target both 777:2
```

Example:
Hub-and-Spoke VPN

In one type of a hub-and-spoke design, only the hub site can reach every other site in the VPN. All other sites—spokes—can reach only the hub site. (More complex hub-and-spoke designs are possible, but require additional configuration besides route targets to achieve.) In Figure 3-11, Customer Site 1 is the hub site for VPN A. As such it can reach both spokes, Customer Sites 2 and 3 through VRF A. Customer Site 2 can reach only the hub, customer 1, through VRF C. Customer Site 3 can reach only the hub, customer 1, through VRF E.

BGP sessions exist between PE 1 and PE 2 and between PE 1 and PE 3. In most situations, BGP itself would be fully meshed, but that level of complexity is not necessary for this example. The MPLS paths through the service provider core are omitted for clarity.

To configure route targets for this hub and spoke, you specify different import and export route targets on the hub VRF. On the spoke VRFs, you switch these route targets.

Route-target configuration on PE 1:

```
host1(config)#virtual-router newyork
host1:newyork(config)#ip vrf vrfA
host1:newyork(config-vrf)#route-target export 777:25
host1:newyork(config-vrf)#route-target import 777:50
```

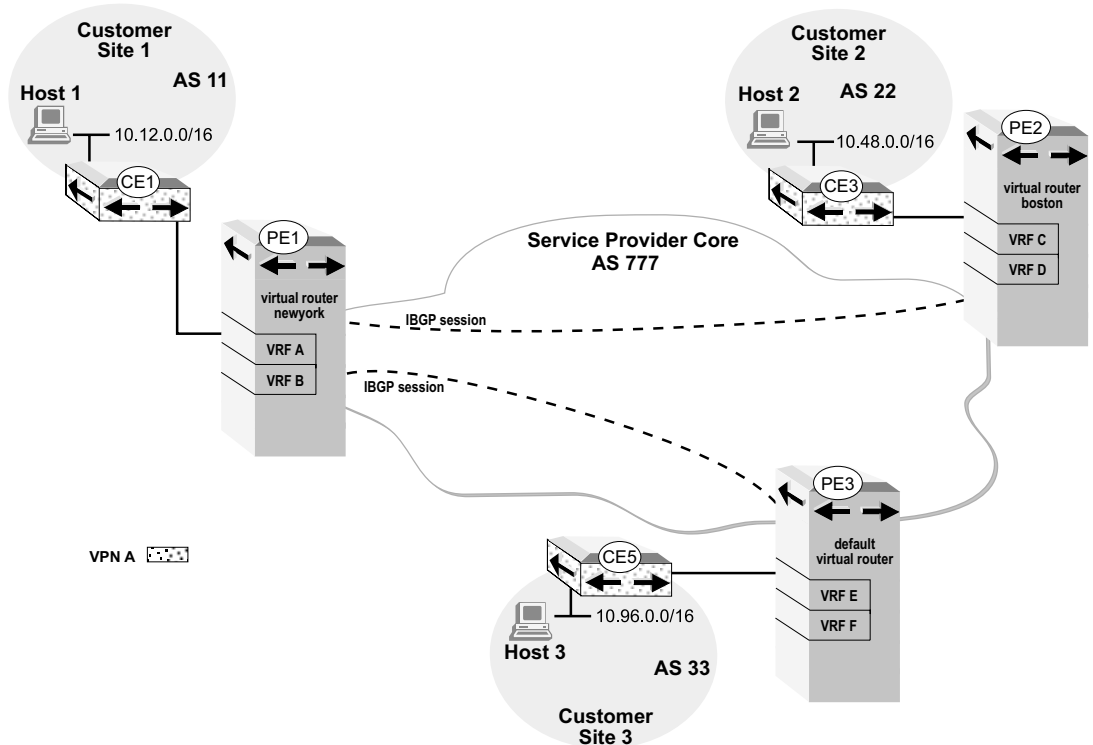


Figure 3-11 Hub-and-spoke VPN

Route-target configuration on PE 2:

```
host2(config)#virtual-router boston
host2:boston(config)#ip vrf vrfC
host2:boston(config-vrf)#route-target export 777:50
host2:boston(config-vrf)#route-target import 777:25
```

Route-target configuration on PE 3:

```
host3(config)#ip vrf vrfE
host3(config-vrf)#route-target export 777:50
host3(config-vrf)#route-target import 777:25
```

This configuration ensures that when VRF E on PE 3 receives an update message from PE 1, BGP installs the advertised route only if it has a route target of 25. Routes from PE 2 have a route target of 50, and cannot be installed. Similarly, when VRF C on PE 2 receives an update message from PE 1, BGP installs the advertised route only if it has a route target of 25. Routes from PE 3 have a route target of 50, and cannot be installed. When PE 1 receives updates from either PE 2 or PE 3, the routes have a

route target of 50, match VRF A's import list, and are installed in VRF A's forwarding table.

Setting Import and Export Maps for a VRF

The combination of the export route target of VRF A and the route-target import list of VRF B determines whether routes from VRF A are distributed to VRF B.

You can associate import maps and export maps with VRFs to provide finer-grained control of route distribution:

- **Export maps** – You can use an export map to change the attributes of a route when it is exported from a VRF. Export maps do not filter routes.
- **Import maps** – You can use an import map to change the attributes of a route when it is imported to a VRF. You can also use an import map to filter routes. If you associate an import map with a VRF, that VRF then accepts only received routes that pass the import map (and match the import route target list).

For information on creating a route map to be used as an import or export map, see *Chapter 1, Configuring BGP Routing*. The following example shows how to apply the route map *A-route-map* to the VRF *vpnA* configured on the virtual router *boston*.

```
host1(config)#virtual router boston
host1:boston(config)#ip vrf vpnA
host1:boston(config-vrf)#import map A-route-map
```

export map

- Use to apply an export route map to a VRF.
- Example

```
host1:boston(config-vrf)#export map A-route-map
```

- Use the **no** version to remove the route map from the VRF.

import map

- Use to apply an import route map to a VRF.
- Example

```
host1:boston(config-vrf)#import map Another-route-map
```

- Use the **no** version to remove the route map from the VRF.

Assigning an Interface to a VRF

You must assign an interface or subinterface to a VRF so that when the system receives a packet at this interface, it routes the packet using the

VRF's forwarding table rather than the global forwarding table. You can assign the interface from outside the context of the VRF or inside the context of the VRF.

To assign an interface to a VRF from outside the VRF context:

1 Select the interface.

2 Specify the VRF to associate with the interface.

```
host1:vr1(config)#interface gigabitEthernet 1/0
host1:vr1(config-if)#ip vrf forwarding vrfA
```

3 Assign an IP address to the interface because forwarding the interface from the VR to the VRF removes the existing IP configuration from the interface.

```
host1:vr1(config-if)#ip address 10.16.2.77 255.255.255.0
```

To assign an interface to a VRF from inside the VRF context:

1 Select the interface.

2 Enter the VRF context.

```
host1:vr1(config)#virtual-router :vrfA
```

3 Associate the interface.

```
host1:vr1:vrfA(config)#interface gigabitEthernet 1/0
```

In this case, you do not have to reassign an IP address to the interface because you did not use the **ip vrf forwarding** command.

ip vrf forwarding

- Use to assign a VRF to an interface or subinterface by forwarding the interface from the VR to the VRF.
- Forwarding the interface removes the IP configuration from the interface. You must reassign an IP address to the interface after you issue this command.
- Example

```
host1:foo(config)#ip vrf forwarding vrfA
host1:foo(config)#ip address 10.12.4.5 255.255.255.0
```

- Use the **no** version to remove the interface assignment.

Adding Static Routes to a VRF

Consider the network structure shown in Figure 3-12. If no routing protocol—BGP or any other IGP—is running between the PE and the CE, you must use the **ip route vrf** command to add a static route in the customer's VRF for each prefix in that customer's site.

Each of these static routes should point to the link connecting the PE to the CE. Typically, you redistribute these static routes in the VRF's address family in BGP or use **network** commands to make those prefixes reachable from other CEs in the same VPN.

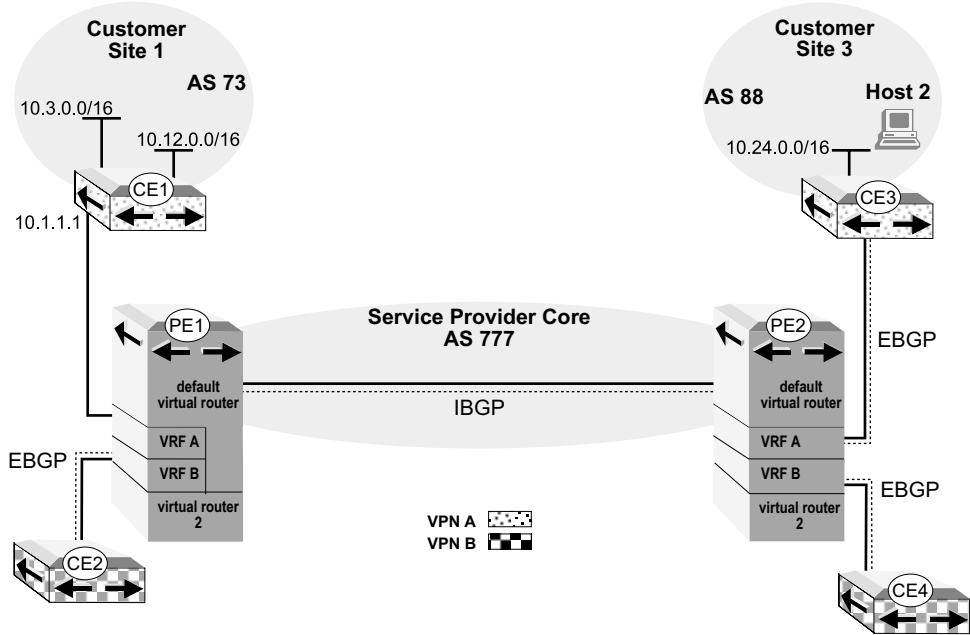


Figure 3-12 Configuring static routes

In Figure 3-12, PE 2 has external BGP connections to CE 3 and CE 4. PE 1 has an EBGP connection to CE 2. However, no BGP (or IGP) connection exists between PE 1 and CE 1. The following example shows how to configure static routes on VRF A for both prefixes in CE 1.

```

host1(config)#virtual-router pe1
host1:pe1(config)#ip vrf vpnA
host1:pe1(config-vrf)#ip route vrf vrfA 10.3.0.0 255.255.0.0
10.1.1.1
host1:pe1(config-vrf)#ip route vrf vrfA 10.12.0.0
255.255.0.0 10.1.1.1

```

ip route vrf

- Use to add a static route to a VRF.
- Example

```

host1:pe1(config-router-af)#ip route vrf vrfA 10.0.0.0
255.0.0.0 192.168.1.1

```

- Use the **no** version to remove a static route from a VRF.

Configuring IGPs on the VRF

If you do not configure static routes on the VRF for each prefix in the associated customer site, then you must configure an IGP on the VRF so that the VRF can learn routes from customer sites.

Configuring the IGP in the VRF Context

After creating a VRF, you can access it as if it were a virtual router for the purpose of configuring the IGP.

If you are in the context of the virtual router that has the VRF, you access the VRF as follows:

```
host1(config)#virtual-router vrfa  
host1:default:vrfa(config)#
```

If you are *not* in the context of the virtual router that has the VRF, you access the VRF as follows:

```
host1(config)#virtual-router boston:vrfa  
host1:boston:vrfa(config)#
```

The following sample commands illustrate one way to configure OSPF; you can configure RIP and IS-IS similarly:

```
host1(config)#ip vrf vrfa  
host1(config-vrf)#rd 100:5  
host1(config-vrf)#route-target both 100:5  
host1(config-vrf)#exit  
host1(config)#virtual-router vrfa  
host1:default:vrfa(config)#router ospf 100  
host1:default:vrfa(config-router)#redistribute bgp
```

At this point you proceed with the IGP configuration for the VRF.

Configuring the IGP outside the VRF Context

The RIP and OSPF protocols also enable you to specify a VRF and configure the protocol without actually entering the VRF context.

For example, for OSPF you could issue the following command and then complete OSPF configuration tasks for VRF A:

```
host1(config)#router ospf 100 vrf vrfa
```

For RIP, you create the RIP process, specify the address family for the VRF, and specify redistribution of BGP routes for VRF A:

```
host1(config)#router rip 100  
host1(config-router)#address-family ipv4 vrf vrfa  
host1(config-router-af)#redistribute bgp
```

At this point you proceed with RIP configuration for the VRF.

See the appropriate chapter for information on configuring the desired IGP:

- *ERX Routing Protocols Configuration Guide, Vol. 1, Chapter 6, Configuring RIP*
- *ERX Routing Protocols Configuration Guide, Vol. 1, Chapter 7, Configuring OSPF*
- *ERX Routing Protocols Configuration Guide, Vol. 1, Chapter 8, Configuring IS-IS*
- *ERX Routing Protocols Configuration Guide, Vol. 2, Chapter 1, Configuring BGP Routing*

virtual-router

- Use to access a VRF to configure it with an IGP to learn routes from a CE.
- To access the VRF from its VR context (in this example, the default VR):


```
host1(config)#virtual-router :vrfsouthie
host1:default:southie(config)#
```
- To access the VRF from the context of a different VR:


```
host1(config)#virtual-router boston:southie
host1:boston:southie(config)#
```
- Issuing a **no** version of this command (**no virtual-router** : *vrfName* or **no virtual-router** *vrfName*: *vrfName*) that specifies an existing VRF only displays the error message: “Cannot delete a VRF with this command”; you must use the **no ip vrf** command to remove a VRF.

Disabling Automatic Route-Target Filtering

When BGP receives a VPN-IPv4 route from another PE, BGP stores that route in its local-RIB only if at least one VRF imports a route target of that route. If there is no VRF that imports any of the route targets of the route, BGP discards the route; this feature is called automatic route-target filtering. The intention is that BGP only keeps track of routes for directly connected VPNs, and discards all other VPN-IPv4 routes to conserve memory.

If a new VPN is connected to the system (that is, if the import route-target list of a VRF changes), BGP automatically sends a route-refresh message to obtain the routes that it previously discarded.

You can use the **no bgp default route-target filter** command to disable automatic route-target filtering globally for all VRFs. However, automatic route-target filtering is always disabled on route reflectors that have at

least one route-reflector client. You cannot enable automatic route-target filtering for such route reflectors.

bgp default route-target filter

- Use to control automatic route-target filtering.
- Route-target filtering is enabled by default; use the **no** version to disable automatic filtering.
- Example

```
host1:vrf1(config-router)#no bgp default route-target filter
```
- Takes effect immediately. If route target filtering is turned on, immediately removes routes that should be filtered.
If route target filtering is turned off, BGP automatically sends out a route-refresh message over every VPNv4 unicast session (for which the route refresh capability was negotiated) to get previously filtered routes. If the route refresh capability was not negotiated over the session, BGP bounces the session.
- Use the **no** version to disable automatic route-target filtering.

Creating Labels Per FEC

By default, the system minimizes the number of stacked labels to be managed by generating a single label for all BGP routes advertised by a given VRF; this is a per-VRF label. Upon receiving traffic for a per-VRF label, the system performs a label pop and a route lookup to forward the traffic to the next hop.

You can use the **ip mpls forwarding-mode label-switched** command to configure the system to generate a label for each different FEC that a BGP route points to in the VPN; this is a per-FEC label. Issuing this command enables you to avoid a route lookup for traffic destined for CEs, because in this mode traffic is label switched to the corresponding next hop over that interface; a route lookup is not performed.

For the following types of routes, the system always generates a per-VRF label and forwards traffic after a route lookup (rather than label switching the traffic without a route lookup) regardless of the status of this command:

- Local connected interfaces redistributed into BGP, regardless of the interface type.
- BGP redistributed routes that point to Ethernet interfaces with a null next-hop address; that is, when the next-hop address is the local address on the Ethernet interface.
- BGP redistributed routes that point to loopback interfaces.

The following commands configure a system where BGP is running in VRF pe11 and static and connected routes are redistributed into the VRF:

```
host1(config)#ip vrf pe11
host1(config-vrf)#ip mpls forwarding-mode label-switched
host1(config-vrf)#ip route vrf pe11 10.3.4.5 255.255.255.255
    fastEthernet 0/1
host1(config-vrf)#ip route vrf pe11 10.1.1.1 255.255.255.255
    loopback 1
host1(config-vrf)#exit
host1(config)#router bgp 100
host1(config-router)#address-family ipv4 unicast vrf pe11
host1(config-router-af)#exit
host1(config-router)#no auto-summary
host1(config-router)#no synchronization
host1(config-router)#redistribute static
host1(config-router)#redistribute connected
```

For each connected route that is redistributed into the VRF and advertised across the BGP/MPLS VPN, the system assigns a per-VRF label rather than a per-FEC label.

The static route 10.3.4.5/32 points to an Ethernet interface where the next-hop address is the local address on that interface. BGP therefore advertises this static route with a per-VRF label. The static route 10.1.1.1/32 points to loopback interface 1. BGP therefore advertise this static route with a per-VRF label.

ip mpls forwarding-mode label-switched

- Use to generate a label for each different FEC pointed to by a BGP route.
- For some types of routes, issuing this command has no effect on the labels created; they are always per-VRF labels.
- Example

```
host1:vrl(config-vrf)#ip mpls forwarding-mode label-switched
```

- Use the **no** version to restore the default, generating a single label for all BGP routes sent from a given VRF.

Allocating VPN Interfaces per Label

If you want to observe statistics on a per-label basis, you can use the **ip mpls vpn-interface per-label** command to configure the system to allocate a VPN interface for each received stacked label. Doing so will reduce the number of supported egress FECs to the order of thousands.



Note: If you want to run IP multicasting over MPLS tunnels in a VRF, you must issue the **ip mpls vpn-interface per-label** command. IP multicasting over MPLS

tunnels in a VRF is not supported in the default mode whereby MPLS allocates VPN interfaces per next hop.

You can use the **show ip route** command to observe the received stacked label associated with received BGP routes. In the following example, several routes have been received in VRF pe11:

```
host1:pe1:pe11#show ip route
```

```
Protocol/Route type codes:
```

```
I1- ISIS level 1, I2- ISIS level2,
I- route type intra, IA- route type inter, E- route type external,
i- metric type internal, e- metric type external,
O- OSPF, E1- external type 1, E2- external type2,
N1- NSSA external type1, N2- NSSA external type2
L- MPLS label, V- VRF
```

Prefix/Length	Type	Next Hop	Dist/Met	Intf
10.21.21.0/24	Bgp	2.2.2.2[L:21]	200/0	ip dyn-0[tun mpls:vpnIngress-33]
10.22.22.0/24	Bgp	2.2.2.2[L:21]	200/0	ip dyn-0[tun mpls:vpnIngress-33]
10.99.99.21/32	Bgp	2.2.2.2[L:26]	200/0	ip dyn-0[tun mpls:vpnIngress-33]

All three routes were received from the same endpoint, indicated by the common BGP indirect next hop, 2.2.2.2. The received stacked label for each route is presented in the Next Hop column as [L: *labelNumber*]. Routes 10.21.21.0/24 and 10.22.22.0/24 have the same label, 21. Route 10.99.99.21/32 has a different label, 26.

In the default mode, a single VPN interface is created for each next hop. That is, a single dynamic IP interface is created on top of the variable interface created by MPLS. All three routes, even though they have different received stacked labels, point to the same dynamic IP interface created for the next hop, ip dyn-0[tun mpls:vpnIngress-33].

Alternatively, if you issue the **ip mpls vpn-interface per-label** command in the same network to create per-label interfaces, the output would look as follows:

```
host1:pe1:pe11#show ip route
```

```
Protocol/Route type codes:
```

```
I1- ISIS level 1, I2- ISIS level2,
I- route type intra, IA- route type inter, E- route type external,
i- metric type internal, e- metric type external,
O- OSPF, E1- external type 1, E2- external type2,
N1- NSSA external type1, N2- NSSA external type2
L- MPLS label, V- VRF
```

Prefix/Length	Type	Next Hop	Dist/Met	Intf
10.21.21.0/24	Bgp	2.2.2.2[L:21]	200/0	ip dyn-0[tun mpls:vpnIngress-33]
10.22.22.0/24	Bgp	2.2.2.2[L:21]	200/0	ip dyn-0[tun mpls:vpnIngress-33]
10.99.99.21/32	Bgp	2.2.2.2[L:26]	200/0	ip dyn-0[tun mpls:vpnIngress-36]

Routes 10.21.21.0/24 and 10.22.22.0/24 have the same received stacked label, 21, and point to the same dynamic IP interface created for that label, ip dyn-0[tun mpls:vpnIngress-33]. Route 10.99.99.21/32 has a different label, 26, so a different dynamic IP interface is created for it, ip dyn-0[tun mpls:vpnIngress-36].

ip mpls vpn-interface per-label

- Use to create a VPN interface for each received stacked label in a BGP/MPLS VPN.
- Enables collection of statistics on a per-label basis.
- Example


```
host1:vrl(config-vrf)#ip mpls vpn-interface per-label
```
- Use the **no** version to restore the default, creating a VPN interface for each next-hop PE.

Configuring PE-to-PE LSPs

See *Chapter 2, Configuring MPLS*, for information on configuring LSPs.

Enabling BGP Routing

You must enable the BGP routing process on the system serving as the PE router.

router bgp

- Use to enable the BGP routing protocol and to specify the local AS—the AS to which this BGP speaker belongs.
- All subsequent BGP configuration commands are placed within the context of this router and AS; you can have only a single BGP instance per virtual router.
- Specify only one BGP AS per virtual router.
- Example


```
host1:vrl(config)#router bgp 100
```
- This command takes effect immediately.
- Use the **no** version to remove the BGP process.

Enabling VPN Address Exchange

To limit the exchange of routes to those from within the VPN-IPv4 address family, and to set other desired BGP parameters:

- 1 Specify that the router should exchange addresses within a VPN by choosing the VPN-IPv4 address family.
- 2 Specify individual neighbors or peer groups that will exchange routes only from within the current (VPN-IPv4) address family.
- 3 Configure BGP parameters for VPN services.

See *Chapter 1, Configuring BGP Routing*, for information on configuring BGP sessions. The section *Understanding BGP Command Scope* in that chapter has tables that list BGP commands according to their scope. From Address Family Configuration mode, you can issue the commands in Table 1-4 and Table 1-6.

- 4 Exit Address Family Configuration mode.

address-family

- Use to configure the router to exchange IPv4 addresses in VPN mode.
- The default setting is to exchange IPv4 addresses in unicast mode from the default router.
- Example

```
host1:vr1(config-router)#address-family vpnv4
```
- This command takes effect immediately.
- Use the **no** version to disable the exchange of a type of prefix.

exit-address-family

- Use to exit Address Family Configuration mode and access Router Configuration mode.
- Example

```
host1:vr1(config-router-af)#exit-address-family
```
- There is no **no** version.

neighbor activate

- Use to specify neighbors that will exchange routes from within the current address family.
- Example

```
host1:vr1(config-router-af)#neighbor 192.168.1.158 activate
```
- Takes effect immediately. If dynamic capability negotiation was not negotiated with the peer, the session is automatically bounced so that the exchanged address families can be renegotiated in the open messages when the session comes back up.

If dynamic capability negotiation was negotiated with the peer, BGP sends a capability message to the peer to advertise or withdraw the multiprotocol capability for the address family in which this command is issued. If a neighbor is activated, BGP also sends the full contents of the BGP RIB of the newly activated address family.

- Use the **no** version to indicate that routes of the current address family should not be exchanged with the peer. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

Configuring PE-to-CE BGP Sessions

If you have established a BGP session between a PE and a particular CE, you can configure BGP sessions with all the other customer sites within the VPN so that they can learn the routes to the particular CE.

Configuring the PE-to-CE external BGP session is a bit different from the usual external BGP session. You must configure the session in the context of the IPV4 unicast address family of the VRF. Consider the topology shown in Figure 3-13.

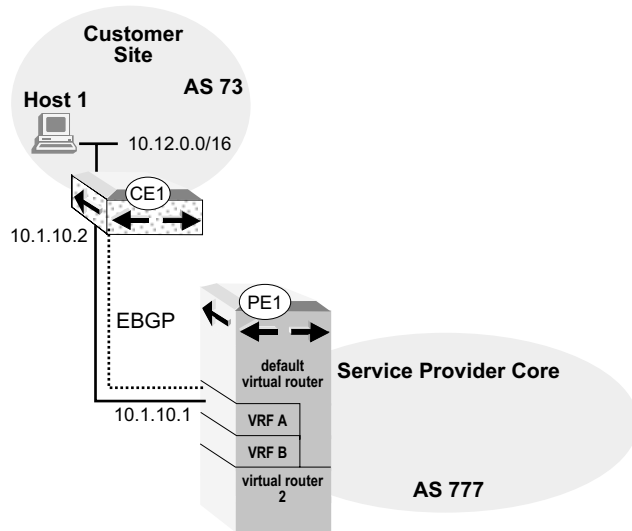


Figure 3-13 PE-to-CE session

You configure the characteristics of VRF A, the global BGP attributes, the address family for the session, and BGP attributes relevant to the VRF or address family.

```
host1(config)#ip vrf vrfa
host1(config-vrf)#rd 777:5
host1(config-vrf)#route-target both 777:5
```

```
host1(config-vrf)#exit
host1(config)#interface gigabitEthernet 1/0
host1(config-if)#ip vrf forwarding vrfA
host1(config-if)#ip address 10.1.10.1 255.255.255.0
host1(config-if)#exit
host1(config)#router bgp 777

    (Configure global BGP attributes)

host1(config-router)#address-family ipv4 unicast vrf vrfA
host1(config-router-af)#neighbor 10.1.10.2 remote-as 73

    (Configure BGP attributes relevant to the VRF or the
    address family)
```

See *Chapter 1, Configuring BGP Routing*, for more information on configuring BGP.

Advertising Static Routes to Customers

If you established static routes on a PE for each prefix in a particular customer site, you can configure BGP on the PE to advertise these static routes to customer sites within the VPN via **network** commands.

```
host1:vr1(config-router)#network 10.3.0.0
host1:vr1(config-router)#network 10.12.0.0
```

In this example, both networks end on a classful boundary, eliminating the need to configure a network mask.

Alternatively, you can use the **redistribute** command to advertise the static routes as follows:

```
host1:vr1(config-router)#redistribute static
```

See *Chapter 1, Configuring BGP Routing*, for more information on advertising static routes.

Advertising IGP Routes to Customers

If the PE learns routes from a CE via an IGP, you can configure BGP to advertise these IGP routes to all customer sites within the VPN via **redistribute** commands. For example, if the PE learns the routes via OSPF, you can issue the following command to inject these routes into BGP for advertisement:

```
host1:vr1(config-router)#redistribute ospf
```

See *Chapter 1, Configuring BGP Routing*, for more information on advertising IGP routes.

Disabling the Default Address Family

PEs can exchange routes in the IPv4 address family, VPNv4 address family, or both. Issuing the **neighbor remote-as** command automatically activates the IPv4 unicast address family, meaning that the PE exchanges routes in the IPv4 unicast address family with that peer.

Example 1 The following commands illustrate how you could configure the exchange of routes in both the IPv4 unicast and the VPNv4 unicast address families for a BGP peer:

```
host1:vr1(config)#router bgp 777
host1:vr1(config-router)#neighbor 10.26.5.10 remote-as 100
host1:vr1(config-router)#address-family vpnv4 unicast
host1:vr1(config-router-af)#neighbor 10.26.5.10 activate
host1:vr1(config-router-af)#exit-address-family
```

The **neighbor remote-as** command activated the IPv4 unicast address family for the peer. The **address-family** command entered the context of the VPNv4 unicast family and the **neighbor activate** command activated the address family for the peer.

Example 2 The following commands illustrate one way you could disable the exchange of routes in the IPv4 unicast address family and enable the exchange of routes in the VPNv4 unicast address family:

```
host1:vr1(config)#router bgp 777
host1:vr1(config-router)#neighbor 10.26.5.10 remote-as 100
host1:vr1(config-router)#address-family ipv4 unicast
host1:vr1(config-router-af)#no neighbor 10.26.5.10 activate
host1:vr1(config-router-af)#exit-address-family
host1:vr1(config-router)#address-family vpnv4 unicast
host1:vr1(config-router-af)#neighbor 10.26.5.10 activate
host1:vr1(config-router-af)#exit-address-family
```

In this case, the **no neighbor activate** command specifically disables the IPv4 unicast address family for that peer alone; no other peers are affected. The VPNv4 unicast address family is activated for the peer as in Example 1.

Example 3 The following commands illustrate another way you could disable the exchange of routes in the IPv4 unicast address family and enable the exchange of routes in the VPNv4 unicast address family:

```
host1:vr1(config)#router bgp 777
host1:vr1(config-router)#no bgp default ipv4-unicast
host1:vr1(config-router)#neighbor 10.26.5.10 remote-as 100
host1:vr1(config-router)#address-family vpnv4 unicast
host1:vr1(config-router-af)#neighbor 10.26.5.10 activate
```

```
host1:vr1(config-router-af)#exit-address-family
```

In this case, the **no bgp default ipv4-unicast** command prevents the automatic enabling of the IPv4 unicast address family for all peers subsequently configured with the **neighbor remote-as** command. Previously configured peers are not affected. The VPNv4 unicast address family is activated for the peer as in Examples 1 and 2.

Using a Single ASN for all CE Sites

If you want to use the same AS number for all of your CE sites, you can substitute a PE's autonomous system number for that of a neighbor by specifying the neighbor's IP address in the **neighbor as-override** command. If you fail to do this, the CE recognizes its AS in the AS path of received routes and believes it has discovered a routing loop; the routes will be rejected.

Example In the following example, the router's AS number of 777 overrides the neighboring router's AS number of 100.

```
host1:vr1(config)#router bgp 777
host1:vr1(config-router)#neighbor 172.16.20.10 remote-as 100
host1:vr1(config-router)#neighbor 172.16.20.10 update-source
loopback0
host1:vr1(config-router)#address-family ipv4 vrf vpn1
host1:vr1(config-router-af)#neighbor 172.25.14.12 remote-as
100
host1:vr1(config-router-af)#neighbor 172.25.14.12
as-override
```

neighbor as-override

- Use to enable the use of the same AS number for all CE sites by substituting the current router's AS number in routing tables for that of the neighboring router.
- If you specify a BGP peer group by using the *peer-group-name* argument, all the members of the peer group inherit the characteristic configured with this command. You cannot override the characteristic for a specific member of the peer group.

- Example

```
host1:vr1(config-router)#neighbor 192.168.255.255
as-override
```

- Example

```
host1(config-router)#neighbor 192.168.1.158 as-override
```

- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

To apply the new policy to routes that are already present in the BGP RIB, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

- Use the **no** version to halt the substitution of the AS numbers. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

Advertising Prefixes with Duplicate AS Numbers

When a BGP speaker receives a route that has the speaker's AS number in its AS path, the speaker declares that route to be a loop and discards it. However, in some circumstances, as in the implementation of a hub-and-spoke VPN topology, this is not the desired behavior. You want the BGP speaker (hub) to accept such routes. You can use the **neighbor allows-in** command to specify the number of times that a route's AS path can contain the BGP speaker's AS number.

neighbor allows-in

- Use to enable the acceptance of all routes whose AS path contains the BGP speaker's AS number up to the specified number of times.
- If the AS path of a route contains the speaker's AS number more than the specified number of times, the route is considered to be a loop and is discarded.
- Example

```
host1(config-router)#bgp allows-in
```

- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

To apply the new policy to routes that are already present in the BGP RIB, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

- Use the **no** version to prevent the acceptance of these routes, resulting in the BGP speaker's discarding the routes.

Controlling Route Importation

You can control how many routes a PE can add to a particular VRF's forwarding table by specifying a maximum limit and a warning threshold. When the router attempts to add a route, it checks the limit you configure against a route count it maintains for routes already in the VRF's forwarding table.

With a warning threshold configured, the following behavior takes place when the PE attempts to add a route:

- When adding the route causes the route count to exceed the warning threshold for the first time, the system adds the route and generates a warning-threshold-exceeded log entry.
- As long as the route count stays above the warning threshold, adding more routes does not generate more warning-threshold-exceeded log entries.
- If the route count fluctuates below and above the warning threshold due to route deletions and additions, an interval of five minutes since the last warning-threshold-exceeded log entry must pass before another warning-threshold-exceeded log entry can be generated. This behavior prevents the system log from being flooded with log entries.

With a limit configured, the following behavior takes place when the PE attempts to add a route:

- When adding the route causes the route count to exceed the limit for the first time, the system rejects the route and generates a limit-exceeded log entry.
- As long as the route count stays at the limit, further attempts to add routes fail, but do not generate any more limit-exceeded log entries.
- If the route count fluctuates below and up to the limit due to route deletions and additions, no further limit-exceeded log entries are generated until a five minute interval has passed since the last limit-exceeded log entry. This behavior prevents the system log from being flooded with log entries.

When you issue the command, the system immediately reevaluates the current number of routes against the new limit. If the current number of routes is greater than the maximum allowed, the system might remove dynamically learned routes in order to enforce the new limit.

maximum routes

- Use to prevent a PE router from importing too many routes from attached CE routers into a particular VRF.
- When the router attempts to add a route that exceeds the *warningThreshold*, the system generates a warning-threshold log entry and adds the route. An interval of five minutes must pass before another warning-threshold-exceeded message can be generated.
- When the router attempts to add a route that exceeds the *limit*, the system generates a limit-exceeded warning and rejects the route. An interval of five minutes must pass before another limit-exceeded message can be generated.
- Messages are logged to ipRouteTable at severity error.
- The interval timers for the limit and the warning threshold are independent.
- You can use the **warning-only** keyword to specify that the system add the route and generate a warning-threshold-exceeded log entry (instead of a limit-exceeded log entry) when the limit is exceeded.
- Issuing the command causes the system to evaluate the current route count and determine whether to generate new messages; any existing warning threshold or limit timers are reset to zero.
- Example

```
host1(config-vrf)#maximum routes 80 65
```
- Use the **no** version to remove the limit and warning threshold.

Deleting Routes for a VRF

You can delete one or all IP routes assigned to a VRF or all VRFs.

clear ip routes

- Use to clear routes from the routing table of one or all VRFs.
- If you do not specify a VRF, routes are removed from all VRFs.
- You can specify either that a single route or all dynamic routes are to be removed.
- Example

```
host1:vr1#clear ip routes vrf vr3 *
```
- This command takes effect immediately.
- There is no **no** version.

OSPF and BGP/MPLS VPNs

Before reading this section, you should be thoroughly familiar with OSPF. For detailed information on that protocol, see *ERX Routing Protocols Configuration Guide, Vol. 1, Chapter 7, Configuring OSPF*.

You can use BGP/MPLS VPNs to connect OSPF domains without creating OSPF adjacencies between the domains. The BGP/MPLS VPN

backbone acts as either an OSPF backbone (area 0) or an OSPF area above the backbone.

In this topology, OSPF is the routing protocol between the CE and the PE. This OSPF link can be configured in area 0 or any other OSPF area. However, if the customer site has any connections to area 0, then at least one OSPF router configured on a CE must have an area 0 link to a PE site. In this case, the BGP/MPLS VPN acts as if it is in an area above the OSPF backbone area. When the PE-CE link is in a nonbackbone area, the BGP/MPLS VPN acts as an OSPF backbone.

In either case, the OSPF router configured as a PE in the BGP/MPLS VPN is always treated as an area border router (ABR) and functions as an area 0 router so that it can distribute interarea routes to the CE. The BGP/MPLS VPN distributes both interarea and intra-area routes between PEs as inter-area, type 3 summary routes.

Distributing OSPF routes from CE to PE

You configure OSPF in the VRF associated with the VPN and associate the interface connected to the CE with the VRF. OSPF routes can then propagate from a CE to a PE when an OSPF adjacency has formed between the two routers. OSPF adds routes to the VRF's forwarding table at the PE side with routes learned from the CE.

Distributing routes Between PEs

The OSPF routes in the VRF forwarding table are OSPF IPv4 routes, but BGP/MPLS VPNs distribute VPN-IPv4 routes via MP-BGP. You must configure the VRF to redistribute the OSPF routes into MP-BGP. MP-BGP converts each imported OSPF route to a VPN-IPv4 route, applies export policy to the route, and then propagates the route to a remote PE site via the MPLS/VPN backbone. At the destination PE, MP-BGP places each route in the appropriate VRF forwarding table based on the import policy for each VRF and the route target associated with the route.

Preserving OSPF Routing Information Across the MPLS/VPN Backbone

MP-BGP attaches two new extended community attributes to the routes redistributed from OSPF:

- OSPF domain identifier extended community attribute
- OSPF route type extended community attribute

MP-BGP uses these attributes and the MED to preserve OSPF routing information across the BGP/MPLS VPN backbone.

OSPF Domain Identifier Attribute

The OSPF domain identifier attribute uniquely identifies the OSPF domain from which a route was redistributed into MP-BGP.

You must configure an OSPF domain ID for the VRFs on the PE with the **domain-id** command. All VRFs that belong to a given OSPF domain must be configured with the same domain ID. If not configured, the domain ID defaults to zero. If you configure a value of zero, MP-BGP does not attach an OSPF domain identifier attribute.

If the OSPF domain ID for the destination PE differs from the originating PE, MP-BGP redistributes the route into OSPF as an OSPF type 5 external route.

OSPF Route Type Attribute

The route type attribute carries the OSPF area ID and LSA type:

Type of route	Origin of route
1 – intra-area route	Type 1 LSA
2 – intra-area route	Type 2 LSA
3 – interarea summary route	Type 3 LSA
5 – external route (area ID = 0)	Type 5 LSA
7 – external route (area ID = 0)	Type 7 LSA

MP-BGP uses the route type conveyed by this extended community attribute to determine the best OSPF route when it installs the routes in the VRF forwarding table on the destination PE.

Distributing OSPF Routes from PE to CE

At the remote PE site, MP-BGP converts the OSPF routes to BGP VPN-IPv4 routes and sends them across the BGP/MPLS VPN backbone. At the destination PE, MP-BGP must redistribute the BGP VPN-IPv4 routes back into OSPF IPv4 routes. The PE OSPF router becomes the originator of the routes, which are either type 5 external routes or type 3 internal routes. The PE can announce the OSPF routes to the appropriate CE router via its directly connected PE-CE OSPF link.

If the route has a route type of inter or intra, it is redistributed as a type 3 summary interarea route and the destination PE router generates a type 3 LSA for it.

A route is redistributed as an external route if the route:

- Originates in an OSPF domain that is different from that of the destination PE router.
- Has a route type of 5 or 7, both of which indicate an external route.

In the first case, the PE advertises the route as an external type 2 route. In the second case, the PE advertises the route as an external type 2 route if the least-significant bit is set in the option byte in the route type extended community attribute; otherwise the PE advertises the route as external type 1 route.

Preventing Routing Loops

PE routes ignore OSPF routes received from a CE if the routes are advertised by:

- A type 3 LSA with the most-significant bit set in the LSA options field.
- A type 5 LSA that has a tag value equal to the VPN route tag associated with the OSPF VRF on that PE.

When the destination PE router originates a type 3 LSA learned from BGP to a CE, the PE sets the most-significant bit in the LSA options field to identify the LSA as being generated from a PE. This prevents the LSA from being passed back to the BGP/MPLS VPN through a different PE.

When the destination PE router originates a type 5 LSA learned from BGP to a CE, the PE replaces the external route tag in the LSA with the VPN route tag. You configure the VPN route tag for the OSPF VRF on the PE with the **domain-tag** command. The value of a VPN route tag must be unique within an OSPF domain, so that the same external route is not propagated back to the BGP/MPLS VPN backbone through another PE-CE link.

Configuration Tasks

At a minimum, perform the following tasks on each PE to configure them for OSPF. For other OSPF configuration tasks, see *ERX Routing Protocols Configuration Guide, Vol. 1, Chapter 7, Configuring OSPF*.

- 1 Create the VRF.

```
host1(config)#ip vrf ospf2
Proceed with new VRF creation? [confirm]
```

```
host1(config-vrf)#rd 100:85  
host1(config-vrf)#exit
```

- 2 Start OSPF on the VRF, either from the parent VR or directly from the VRF.

From the parent VR:

```
host1(config)#router ospf 5 vrf ospf2
```

From the VRF:

```
host1(config)#virtual-router :ospf2  
host1:default:ospf2(config)#router ospf 5
```

The command prompts in the remaining steps reflect using the latter method for starting OSPF.

- 3 Configure the OSPF domain ID.

```
host1:default:ospf2(config-router)#domain-id 45
```

- 4 Configure the VPN route tag.

```
host1:default:ospf2(config-router)#domain-tag 1200
```

- 5 Redistribute routes learned from other PEs back into OSPF.

```
host1:default:ospf2(config-router)#redistribute bgp
```

- 6 Create an address family in BGP.

```
host1:default(config)#router bgp 100  
host1:default(config-router)#address-family ipv4 unicast vrf  
ospf2
```

- 7 Redistribute OSPF routes into BGP.

```
host1:default(config-router)#redistribute ospf
```

domain-id

- Use to set the OSPF domain ID for an OSPF VRF on a PE router; the default value is zero.
- Use the same domain ID for all OSPF VRFs in a given OSPF domain.
- When the value is zero, MP-BGP does not attach an OSPF domain identifier attribute when it converts an OSPF route to an MP-BGP route to cross the BGP/MPLS VPN.
- Example

```
host1:default:ospf2(config-router)#domain-id 45
```
- Use the **no** version to restore the default value.

domain-tag

- Use to set the VPN route tag for an OSPF VRF on a PE router.
- The default value is a 32-bit number based on the AS number of the BGP/MPLS VPN backbone, with the first 16 bits set to 1110 0000 0000 0000, followed by the 16 bits representing the AS number.
- Example

```
host1:default:ospf2(config-router)#domain-tag 1200
```
- Use the **no** version to restore the default value.

Monitoring BGP/MPLS VPNs

To view BGP/MPLS VPN settings, you can issue the following **show** commands as well as any of the **show ip bgp** commands described in *Chapter 1, Configuring BGP Routing*. Refer to *Chapter 2, Configuring MPLS*, for information on **show** commands to monitor MPLS settings.

Use the **debug ip mbgp** command to get information on problems with BGP or the network.

debug ip mbgp

- Use to display information on MP-BGP logs for inbound or outbound events, or both.
- Example

```
host1#debug ip mbgp
```
- There is no **no** version, but you can use the **undebug ip mbgp** command to disable display of information previously enabled with the **debug ip mbgp** command.

show ip bgp next-hops

- Use to display information about BGP next hops.
- Specify all VRFs or a particular VRF, and all indirect next hops or a particular indirect next hop.
- Field descriptions
 - › Indirect next-hop – BGP next-hop attribute as received in the BGP update message
 - › MPLS stacked label – MPLS label as received in the MP-Reach-NLRI attribute in the BGP update message; shown only for VPN routes
 - › Reachable – indicates whether or not the indirect next-hop is reachable. For non-VPN routes, the indirect next-hop is reachable if it is resolved by a route in the IP forwarding table. For VPN routes, the indirect next-hop is reachable if an MPLS base tunnel to the indirect next-hop exists and MPLS successfully created a stacked tunnel on top of that base tunnel using the MPLS stacked label.

- › Direct next-hop – interface and next-hop IP address which resolve the indirect next-hop
- › Reference count – number of routes using this next-hop
- Example

```

host1:pe1#show ip bgp vpnv4 vrf pe11 next-hops
Indirect next-hop 2.2.2.2
  MPLS stacked label 19
  Reachable
  Direct next-hop tun mpls:vpnInL19-18
  Reference count is 1

Indirect next-hop 11.11.11.2
  Reachable (metric 1)
  Direct next-hop atm2/0.11 (11.11.11.2)
  Reference count is 1

```

show ip interface vrf

- Use to display information about the interfaces associated with the specified VRF.
- Field descriptions
 - › interface – interface type and interface specifier
 - › interface status – status of the interface
 - › line protocol – status of the line protocol
 - › Link up/down trap – status of SNMP link up/down traps on the interface
 - › Internet address – IP address of the interface
 - › Operational MTU – actual MTU for the interface
 - › Administrative MTU – configured MTU for the interface
 - › Operational speed – actual speed
 - › Administrative speed – configured speed
 - › Discontinuity Time – value of sysUpTime the last time the integrity of the interface statistics was compromised
 - › Router advertisement – whether routes are advertised; enabled or disabled
 - › Administrative debounce-time – whether the up/down state of the interface will be debounced or *damped* if a link periodically fails and immediately comes back; the time delay (configured in Interface Configuration mode) that an interface must remain in a new state before the routing protocols react to the state change
 - › Operational debounce-time – whether the up/down state of the interface will be debounced or “damped” if a link periodically fails and immediately comes back; the time delay that an interface must remain in a new state before the routing protocols react to the state change; the time delay (configured in Interface Configuration mode or Global Configuration mode) that an interface must remain in a new state before the routing protocols react to the state change
 - › Access routing – when enabled, an access route is installed to the host on the other end of the interface

- › Multipath mode – algorithm used for ECMP, DA/SA hashing or round-robin
- › In Received Packets, Bytes – total packets and bytes received on an IP interface
 - Unicast – unicast packets and bytes received on an IP interface
 - Multicast – multicast packets and bytes received on an IP interface
- › In Policed Packets – packets discarded on a receive IP interface because of token bucket limiting
- › In Error Packets – packets discarded on a receive IP interface because of IP header errors
- › In Invalid Source Address Packets – packets discarded on a receive IP interface because of invalid IP source address (sa-validate enabled)
- › Out Forwarded Packets, Bytes – packets and bytes forwarded out an IP interface
 - Unicast – unicast packets and bytes forwarded out an IP interface
 - Multicast – multicast packets and bytes forwarded out an IP interface
- › Out Scheduler Drops Committed Packets – committed packets dropped because of out queue threshold limit
- › Out Scheduler Drops Conformed Packets – conformed packets dropped because of out queue threshold limit
- › Out Scheduler Drops Exceeded Packets – exceeded packets dropped because of out queue threshold limit
- › Out Policed Packets – packets discarded on a forwarding IP interface because of token bucket limiting
- Example

```
host1#show ip interface vrf vpn1
null0 is up, line protocol is up
  Network Protocols: IP
  Internet address is 255.255.255.255/255.255.255.255
  Broadcast address is 255.255.255.255
  Operational MTU = 1500 Administrative MTU = 0
  Operational speed = 100000000 Administrative speed = 0
  Discontinuity Time = 0
  Router advertisement = disabled
  Administrative debounce-time = disabled
  Operational debounce-time = disabled
  Access routing = disabled
  Multipath mode = hashed

atm4/0.77 is up, line protocol is up
  Network Protocols: IP
  Internet address is 7.8.7.7/255.255.255.0
  Broadcast address is 255.255.255.255
  Operational MTU = 9180 Administrative MTU = 0
  Operational speed = 155520000 Administrative speed = 0
  Discontinuity Time = 0
```

```

Router advertisement = disabled
Administrative debounce-time = disabled
Operational debounce-time = disabled
Access routing = disabled
Multipath mode = hashed

In Received Packets 0, Bytes 0
  Unicast Packets 0, Bytes 0
  Multicast Packets 0, Bytes 0
In Policed Packets 0, Bytes 0
In Error Packets 0
In Invalid Source Address Packets 0
Out Forwarded Packets 0, Bytes 0
  Unicast Packets 0, Bytes 0
  Multicast Packets 0, Bytes 0
Out Scheduler Drops Committed Packets 0, Bytes 0
Out Scheduler Drops Conformed Packets 0, Bytes 0
Out Scheduler Drops Exceeded Packets 0, Bytes 0
Out Policed Packets 0, Bytes 0

```

host1#**show ip interface vrf vpn1 brief**

Interface	IP-Address	Status	Protocol	Description
null0	255.255.255.255	up		up
atm4/0.77	7.8.7.7	up		up

show ip protocols

- Use to display information about the routing protocols associated with the VRF.
- You must specify the name of the VRF for which the protocols are displayed; otherwise, the command displays all protocols configured on the system
- Field descriptions
 - › For BGP:
 - Redistributing – protocol to which BGP is redistributing routes
 - Default local preference – local preference value
 - IGP synchronization – status of IGP synchronization: enabled, disabled
 - Always compare MED – status of multiexit discrimination: enabled, disabled
 - Router flap damping – status of route dampening: enabled, disabled
 - Administrative Distance – external, internal, and local administrative distances
 - Neighbor Address – the IP address of the BGP neighbor
 - Neighbor Incoming/Outgoing update distribute list – number of the access list for outgoing routes
 - Neighbor Incoming/Outgoing update prefix list – number of the prefix list for incoming or outgoing routes

- Neighbor Incoming/Outgoing update prefix tree – number of the prefix tree for incoming or outgoing routes
 - Neighbor Incoming/Outgoing update filter list – number of filter list for incoming routes
 - Routing for Networks – the network for which BGP is currently injecting routes
- › For IS-IS:
- System Id – 6-byte value of the system
 - IS-Type – routing type of the router: Level 1, Level 2
 - Distance – administrative distance for IS-IS learned routes
 - Address Summarization – aggregate addresses defined in the routing table for multiple groups of addresses at a given level or routes learned from other routing protocols
 - Routing for Networks – network for which IS-IS is currently injecting routes
- › For OSPF:
- Router ID – OSPF process ID for the router
 - Distance – administrative distance for OSPF learned routes
 - Redistributing – protocol to which OSPF is redistributing routes
 - Address Summarization – aggregate addresses defined in the routing table for multiple groups of addresses at a given level or routes learned from other routing protocols
 - Routing for Networks – network for which OSPF is currently injecting routes
- › For RIP:
- Router Administrative State – RIP protocol state. Enable means it is allowed to send and receive updates. Disable means that it may be configured but it is *not* allowed to run yet.
 - System Version – RIP versions allowed for sending and receiving RIP updates. The system version is currently set to RIP1, which sends RIP version 1 but will receive version 1 or 2. If the version is set to RIP2, the system will send and receive version 2 only. The default is configured for RIP1.
 - Update interval – current setting of the update timer (in seconds)
 - Invalid after – current setting of the invalid timer (in seconds)
 - hold down time – current setting of the hold down timer (in seconds)
 - flushed interval – current setting of the flush timer (in seconds)
 - Filter applied to outgoing route update – access list applied to outgoing RIP route updates
 - Filter applied to incoming route update – access list applied to incoming RIP route updates
 - Global route map – route map that specifies all RIP interfaces on the system
 - Distance – value added to RIP routes added to the IP routing table. The default is 120.

- Interface – interface type on which RIP protocol is running
- Redistributing – protocol to which RIP is redistributing routes
- Routing for Networks – network for which RIP is currently injecting routes

- Example

```

host1:pe1#show ip protocols vrf pe13
Routing Protocol is "ospf 1" with Router ID 13.13.13.1
  Distance is 110
  Redistributing: bgp
    Address Summarization:
      None
  Routing for Networks:
    13.13.13.0/255.255.255.0 area 0.0.0.0
  
```

show ip route vrf

- Use to display the routing table of the specified VRF.
- Field descriptions
 - › Protocol/Route type codes – type of route
 - › Prefix/Length – network prefix for route in VRF routing table
 - › Type – protocol of route
 - › Next Hop – IP address of the next hop to reach route
 - › Dist/Met – administrative distance and metric applied to route
 - › Intf – outgoing interface to reach route
- Example

```

host1#show ip route vrf vpn2
Protocol/Route type codes:
I1- ISIS level 1, I2- ISIS level2,
I- route type intra, IA- route type inter, E- route type external,
i- metric type internal, e- metric type external,
O- OSPF, E1- external type 1, E2- external type2,
N1- NSSA external type1, N2- NSSA external type2
  
```

Prefix/Length	Type	Next Hop	Dist/Met	Intf
45.5.5.5/32	Connect	45.5.5.5	0/1	fastEthernet3/0
56.5.5.0/24	Connect	56.5.5.5	0/1	atm4/0.21

show ip vrf

- Use to display brief information about the VRFs in this virtual router: the route target of each VRF and the interfaces attached to each VRF.
- Specify the VRF name to display the brief information only about that VRF. You must be within the context of the virtual router to which the VRF belongs.
- Field descriptions
 - › VRF Name – name of each VRF
 - › Default RD – default route distinguisher for the VRF

› Interfaces – interfaces configured for the VRF

- Examples:

```
host1#show ip vrf
VRF Name      Default RD      Interfaces
vpn1          1:1             null0
               atm4/0.77
vpn2          1:3             null0
               fastEthernet3/0
               atm4/0.21

host1#show ip vrf vpn1
VRF Name      Default RD      Interfaces
vpn1          1:1             null0
               atm4/0.77
```

show ip vrf detail

- Use to display detailed information about the VRFs in this virtual router.
- Specify the VRF name to display the brief information only about that VRF. You must be within the context of the virtual router to which the VRF belongs.
- Field descriptions
 - › VRF – name of the VRF
 - › Default RD – default route distinguisher for the VRF
 - › VRF IP Router ID – IP address that uniquely identifies the router
 - › Default TTL – time to live value in the IP header
 - › Reassemble Timeout – value to time out reassembled packets
 - › Interface Configured – interface configured for the VRF
 - › Import VPN Route Target Extended Communities – list of VPNs from which the VRF accepts routing information
 - › Export VPN Route Target Extended Communities – list of VPNs to which the VRF sends update messages
 - › Import Route-map – route map associated with the VRF that filters routes received by the VRF
 - › Export Route-map – route map associated with the VRF that filters routes forwarded by the VRF
- Example

```
host1#show ip vrf detail
VRF vpn1; Default RD 1:1
  VRF IP Router ID 10.1.1.1
  Default TTL: 127
  Reassemble Timeout: 30
  Interface Configured:
    null0 atm4/0.77
  Import VPN Route Target Extended Communities:
    1:2
  Export VPN Route Target Extended Communities:
```

```

1:1 1:2
Import Route-map : map2
Export Route-map : map1
VRF vpn2; Default RD 1:3
Interface Configured:
  null0 fastEthernet3/0 atm4/0.21
Import VPN Route Target Extended Communities:
  3:3 10.4.3.0:1
Export VPN Route Target Extended Communities:
  10.4.3.0:1
Import Route-map : map2
No Export Route-map

```

show ip vrf interfaces

- Use to display summary information about all interfaces associated with all VRFs configured in a virtual router.
- Use the **detail** keyword to display detailed information about the interfaces.
- Field descriptions
 - › Interface – interface type and interface specifier
 - › IP-Address – IP address of the interface
 - › Status – status of the interface
 - › Protocol – status of the line protocol
 - › VRF – name of the VRF with which the interface is associated
 - › interface status – status of the interface
 - › line protocol – status of the line protocol
 - › Link up/down trap – status of SNMP link up/down traps on the interface
 - › Internet address – IP address of the interface
 - › IP Statistics Rcvd:
 - local destination – frames with this router as their destination
 - hdr errors – number of packets containing header errors
 - addr errors – number of packets containing addressing errors
 - unkn proto – number of packets received containing unknown protocols
 - discards – number of discarded packets
 - › IP Statistics Frags:
 - reasm ok – number of reassembled packets
 - reasm req – number of requests for reassembly
 - reasm fails – number of reassembly failures
 - frag ok – number of packets fragmented successfully
 - frag creates – number of frames requiring fragmentation
 - frag fails – number of packets unsuccessfully fragmented
 - › IP Statistics Sent:
 - generated – number of packets generated
 - no routes – number of packets that could not be routed

- discards – number of packets that could not be routed that were discarded
- › ICMP Statistics Rcvd:
 - errors – number of error packets received
 - dst unreachable – number of packets received with destination unreachable
 - time exceed – number of packets received with time-to-live exceeded
 - param probs – number of packets received with parameter errors
 - src quench – number of source quench packets received
 - redirect – number of receive packet redirects
 - echo req – number of echo request (PING) packets
 - echo rpy – number of echo replies received
 - timestamp req – number of requests for a timestamp
 - timestamp rpy – number of replies to timestamp requests
 - addr mask req – number of address mask requests
 - addr mask rpy – number of address mask replies
- › ICMP Statistics Sent:
 - errors – number of error packets sent
 - dst unreachable – number of packets sent with destination unreachable
 - time excd – number of packets sent with time-to-live exceeded
 - param probs – number of packets sent with parameter errors
 - src quench – number of source quench packets sent
 - redirect – number of send packet redirects
 - timestamp req – number of requests for a timestamp
 - timestamp rpy – number of replies to timestamp requests
 - addr mask req – number of address mask requests
 - addr mask rpy – number of address mask replies
- › In Received Packets, Bytes – total packets and bytes received on an IP interface
 - Unicast – unicast packets and bytes received on an IP interface
 - Multicast – multicast packets and bytes received on an IP interface
- › In Forwarded Packets, Bytes – packets and bytes forwarded into an output IP interface
- › In Total Dropped Packets, Bytes – total packets and bytes discarded on a receive IP interface
- › In Policed Packets – packets discarded on a receive IP interface because of token bucket limiting
- › In Invalid Source Address Packets – packets discarded on a receive IP interface because of invalid IP source address (sa-validate enabled)
- › In Error Packets – packets discarded on a receive IP interface because of IP header errors
- › In Discarded Packets – packets discarded on the ingress interface because of a configuration problem rather than a problem with the packet itself

- › In Fabric Dropped Packets – packets discarded on a receive IP interface because of internal fabric congestion
- › Out Forwarded Packets, Bytes – packets and bytes forwarded out an IP interface
 - Unicast – unicast packets and bytes forwarded out an IP interface
 - Multicast – multicast packets and bytes forwarded out an IP interface
- › Out Requested Packets, Bytes – packets and bytes requested to be forwarded out an IP interface
- › Out Total Dropped Packets, Bytes – total packets and bytes dropped by an IP interface on output
- › Out Scheduler Drops Committed Packets, Bytes – committed packets and bytes dropped because of out queue threshold limit
- › Out Scheduler Drops Conformed Packets, Bytes – conformed packets and bytes dropped because of out queue threshold limit
- › Out Scheduler Drops Exceeded Packets, Bytes – exceeded packets and bytes dropped because of out queue threshold limit
- › Out Policed Packets – packets discarded on the egress interface because of token bucket limiting
- › Out Discarded Packets – packets discarded on the egress interface because of a configuration problem rather than a problem with the packet itself
- › Out Fabric Dropped Packets – packets dropped because of internal fabric congestion
- Example

```

host1:PE1#show ip vrf interfaces
Interface                IP-Address                Status Protocol  VRF
null0                    255.255.255.255/32      up          up    pe11
atm4/0.134              4.4.4.2/24              up          up    pe11
null0                    255.255.255.255/32      up          up    pe12
ip0                      6.6.6.8/24              up          up    pe12
null0                    255.255.255.255/32      up          up    pe13
loopback1                7.7.7.2/24              up          up    pe13

```

```

host1:PE1#show ip vrf interfaces detail
null0 is up, line protocol is up
  VRF: pe11
  Link up/down trap is disabled

  Internet address is 255.255.255.255/255.255.255.255
IP statistics:
  Rcvd:  0 local destination
         0 hdr errors, 0 addr errors
         0 unkn proto, 0 discards
  Frags: 0 reasm ok, 0 reasm req, 0 reasm fails
         0 frag ok, 0 frag creates, 0 frag fails
  Sent:  0 generated, 0 no routes, 0 discards
ICMP statistics:

```

```
Rcvd: 0 errors, 0 dst unreachable, 0 time exceed
      0 param probs, 0 src quench, 0 redirect,
      0 echo req, 0 echo rpy
      0 timestamp req, 0 timestamp rpy
      0 addr mask req, 0 addr mask rpy
Sent: 0 errors, 0 dst unreachable, 0 time excd
      0 param probs, 0 src qnch, 0 redirect
      0 timestamp req, 0 timestamp rpy
      0 addr mask req, 0 addr mask rpy
```

```
atm4/0.134 is up, line protocol is up
VRF: pell
Link up/down trap is disabled
```

```
Internet address is 4.4.4.2/255.255.255.0
```

```
IP statistics:
```

```
Rcvd: 0 local destination
      0 hdr errors, 0 addr errors
      0 unkn proto, 0 discards
Frgs: 0 reasm ok, 0 reasm req, 0 reasm fails
      0 frag ok, 0 frag creates, 0 frag fails
Sent: 0 generated, 0 no routes, 0 discards
```

```
ICMP statistics:
```

```
Rcvd: 0 errors, 0 dst unreachable, 0 time exceed
      0 param probs, 0 src quench, 0 redirect,
      0 echo req, 0 echo rpy
      0 timestamp req, 0 timestamp rpy
      0 addr mask req, 0 addr mask rpy
Sent: 0 errors, 0 dst unreachable, 0 time excd
      0 param probs, 0 src qnch, 0 redirect
      0 timestamp req, 0 timestamp rpy
      0 addr mask req, 0 addr mask rpy
```

```
In Received Packets 0, Bytes 0
  Unicast Packets 0, Bytes 0
  Multicast Packets 0, Bytes 0
In Forwarded Packets 0, Bytes 0
In Total Dropped Packets 0, Bytes 0
  In Policed Packets 0
  In Invalid Source Address Packets 0
  In Error Packets 0
  In Discarded Packets 0
  In Fabric Dropped Packets 0
```

```
Out Forwarded Packets 0, Bytes 0
  Unicast Packets 0, Bytes 0
```

```
Multicast Packets 0, Bytes 0
Out Requested Packets 0, Bytes 0
Out Total Dropped Packets 0, Bytes 0
  Out Scheduler Drops Committed Packets 0, Bytes 0
  Out Scheduler Drops Conformed Packets 0, Bytes 0
  Out Scheduler Drops Exceeded Packets 0, Bytes 0
  Out Policed Packets 0
  Out Discarded Packets 0
  Out Fabric Dropped Packets 0
```

show mpls tunnels

- Use to display status and configuration for all tunnels or for a specific tunnel in the current router context.
- A result of Incomplete Configuration in the display indicates either no tunnel endpoint or no label distribution protocol.
- Field descriptions
 - › State – status of tunnel, up or down
 - › Out Label – in the default case for a BGP/MPLS VPN, this will be Variable Interface, which indicates that a packet exiting the interface is going through a variable interface and that one of the labels listed further in the display will be prepended to the packet
 - › Mpls Statistics
 - pkts – number of packets sent across tunnel
 - hcPkts – number of high-capacity (64-bit) packets sent across tunnel
 - octets – number of octets sent across tunnel
 - hcOctets – number of high-capacity (64-bit) octets sent across tunnel
 - errors – number of packets that are dropped for some reason before being sent
 - discardPkts – number of packets that are discarded due to lack of buffer space before being sent
 - › Labels – list of labels associated with the variable interface; one will be selected to be prepended to packets before being sent across tunnel
- Examples

The output varies between the default behavior—when the system creates a VPN interface per next-hop PE—and the behavior resulting when you issue the `ip mpls vpn-interface per-label` command—when the system creates a VPN interface for each received stacked label.

VPN interface per next-hop PE:

```
host12#show mpls tunnels

LSP vpnIngress-21 to 3.3.3.3
State: Up
Out label is Variable Interface
102 pkts, 0 hcPkts, 13464 octets
0 hcOctets, 0 errors, 0 discardPkts
```

```
Labels:  
16 17 18 19
```

VPN interface per received stacked label:

```
host12#show mpls tunnels
```

```
LSP vpnInL16-21 to 3.3.3.3  
State: Up  
Out label is 16 on Label 33  
102 pkts, 0 hcPkts, 13464 octets  
0 hcOctets, 0 errors, 0 discardPkts
```

```
LSP vpnInL17-21 to 3.3.3.3  
State: Up  
Out label is 17 on Label 33  
102 pkts, 0 hcPkts, 13464 octets  
0 hcOctets, 0 errors, 0 discardPkts
```

```
LSP vpnInL18-21 to 3.3.3.3  
State: Up  
Out label is 18 on Label 33  
102 pkts, 0 hcPkts, 13464 octets  
0 hcOctets, 0 errors, 0 discardPkts
```

```
LSP vpnInL19-21 to 3.3.3.3  
State: Up  
Out label is 19 on Label 33  
102 pkts, 0 hcPkts, 13464 octets  
0 hcOctets, 0 errors, 0 discardPkts
```

undebg ip mbgp

- Use to disable the display of information on MP-BGP logs that was previously enabled with the **debug ip mbgp** command.
- Example

```
host1#undebg ip mbgp
```
- There is no **no** version.

