Per-VRF Assignment of BGP Router ID

The Per-VRF Assignment of BGP Router ID feature introduces the ability to have VRF-to-VRF peering in Border Gateway Protocol (BGP) on the same router. BGP is designed to refuse a session with itself because of the router ID check. The per-VRF assignment feature allows a separate router ID per VRF using a new keyword in the existing **bgp router-id** command. The router ID can be manually configured for each VRF or can be assigned automatically either globally under address family configuration mode or for each VRF.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

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

Prerequisites for Per-VRF Assignment of BGP Router ID

Before you configure this feature, Cisco Express Forwarding or distributed Cisco Express Forwarding must be enabled in the network, and basic BGP peering is assumed to be running in the network.

Information About Per-VRF Assignment of BGP Router ID

BGP Router ID

The BGP router identifier (ID) is a 4-byte field that is set to the highest IP address on the router. Loopback interface addresses are considered before physical interface addresses because loopback interfaces are more stable than physical interfaces. The BGP router ID is used in the BGP algorithm for determining the best path to a destination where the preference is for the BGP router with the lowest router ID. It is possible to manually configure the BGP router ID using the **bgp router-id** command to influence the best path algorithm.

Per-VRF Router ID Assignment

In Cisco IOS XE Release 2.1 and later releases, support for configuring separate router IDs for each Virtual Private Network (VPN) routing/forwarding (VRF) instance was introduced. The Per-VRF Assignment of BGP Router ID feature introduces the ability to have VRF-to-VRF peering in Border Gateway Protocol (BGP) on the same router. BGP is designed to refuse a session with itself because of the router ID check. The per-VRF assignment feature allows a separate router ID per VRF using a new keyword in the existing **bgp router-id** command. The router ID can be manually configured for each VRF or can be assigned automatically either globally under address family configuration mode or for each VRF.

Route Distinguisher

A route distinguisher (RD) creates routing and forwarding tables and specifies the default route distinguisher for a VPN. The RD is added to the beginning of an IPv4 prefix to change it into a globally unique VPN-IPv4 prefix. An RD can be composed in one of two ways: with an autonomous system number and an arbitrary number or with an IP address and an arbitrary number.

You can enter an RD in either of these formats:

- Enter a 16-bit autonomous system number, a colon, and a 32-bit number. For example:

45000:3

- Enter a 32-bit IP address, a colon, and a 16-bit number. For example:

192.168.10.15:1

How to Configure Per-VRF Assignment of BGP Router ID

Configuring VRF Instances

Perform this task to configure VRF instances to be used with the per-VRF assignment tasks. In this task, a VRF instance named vrf_trans is created. To make the VRF functional, a route distinguisher is created. When the route distinguisher is created, the routing and forwarding tables are created for the VRF instance named vrf_trans.

Before you begin

This task assumes that you have Cisco Express Forwarding or distributed Cisco Express Forwarding enabled.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip vrf** *vrf-name*
4. **rd** *route-distinguisher*
5. **route-target** {**import** | **both**} *route-target-ext-community*
6. **route-target** {**export** | **both**} *route-target-ext-community*
7. **exit**
8. Repeat Step 3 through Step 7 for each VRF to be defined.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip vrf** *vrf-name* **Example:** `Router(config)# ip vrf vrf_trans`	Defines a VRF instance and enters VRF configuration mode.
Step 4	**rd** *route-distinguisher* **Example:** `Router(config-vrf)# rd 45000:2`	Creates routing and forwarding tables for a VRF and specifies the default RD for a VPN. • Use the *route-distinguisher* argument to specify the default RD for a VPN. There are two formats you can use to specify an RD. For more details, see the "Route Distinguisher" section. • In this example, the RD uses an autonomous system number with the number 2 after the colon.
Step 5	**route-target** {**import** | **both**} *route-target-ext-community* **Example:** `Router(config-vrf)# route-target import 55000:5`	Creates a route-target extended community for a VRF. • Use the **import** keyword to import routing information from the target VPN extended community.

	Command or Action	Purpose
		• Use the **both** keyword to both import routing information from and export routing information to the target VPN extended community. • Use the *route-target-ext-community* argument to specify the VPN extended community.
Step 6	**route-target** {**export** | **both**} *route-target-ext-community* **Example:** `Router(config-vrf)# route-target export 55000:1`	Creates a route-target extended community for a VRF. • Use the **export** keyword to export routing information to the target VPN extended community. • Use the **both** keyword to both import routing information from and export routing information to the target VPN extended community. • Use the *route-target-ext-community* argument to specify the VPN extended community.
Step 7	**exit** **Example:** `Router(config-vrf)# exit`	Exits VRF configuration mode and returns to global configuration mode.
Step 8	Repeat Step 3 through Step 7 for each VRF to be defined.	--

Associating VRF Instances with Interfaces

Perform this task to associate VRF instances with interfaces to be used with the per-VRF assignment tasks. In this task, a VRF instance named vrf_trans is associated with a serial interface.

Make a note of the IP addresses for any interface to which you want to associate a VRF instance because the **ip vrf forwarding** command removes the IP address. Step 8 allows you to reconfigure the IP address.

Before you begin

- This task assumes that you have Cisco Express Forwarding or distributed Cisco Express Forwarding enabled.
- This task assumes that VRF instances have been configured in the Configuring VRF Instances, on page 2.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ip address** *ip-address mask* [**secondary**]
5. **exit**

6. **interface** *type number*
7. **ip vrf forwarding** *vrf-name* [**downstream** *vrf-name2*]
8. **ip address** *ip-address mask* [**secondary**]
9. Repeat Step 5 through Step 8 for each VRF to be associated with an interface.
10. **end**
11. **show ip vrf** [**brief** | **detail** | **interfaces** | **id**] [*vrf-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface loopback0`	Configures an interface type and enters interface configuration mode. • In this example, loopback interface 0 is configured.
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 172.16.1.1 255.255.255.255`	Configures an IP address. • In this example, the loopback interface is configured with an IP address of 172.16.1.1.
Step 5	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**interface** *type number* **Example:** `Router(config)# interface serial2/0/0`	Configures an interface type and enters interface configuration mode. • In this example, serial interface 2/0/0 is configured.
Step 7	**ip vrf forwarding** *vrf-name* [**downstream** *vrf-name2*] **Example:** `Router(config-if)# ip vrf forwarding vrf_trans`	Associates a VRF with an interface or subinterface. • In this example, the VRF named vrf_trans is associated with serial interface 2/0/0. **Note** Executing this command on an interface removes the IP address. The IP address should be reconfigured.

	Command or Action	Purpose
Step 8	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 192.168.4.1 255.255.255.0`	Configures an IP address. • In this example, serial interface 2/0/0 is configured with an IP address of 192.168.4.1.
Step 9	Repeat Step 5 through Step 8 for each VRF to be associated with an interface.	--
Step 10	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 11	**show ip vrf** [**brief** | **detail** | **interfaces** | **id**] [*vrf-name*] **Example:** `Router# show ip vrf interfaces`	(Optional) Displays the set of defined VRFs and associated interfaces. • In this example, the output from this command shows the VRFs that have been created and their associated interfaces.

Examples

The following output shows that two VRF instances named vrf_trans and vrf_users were configured on two serial interfaces.

```
Router# show ip vrf interfaces
Interface          IP-Address      VRF                       Protocol
Serial2            192.168.4.1     vrf_trans                 up
Serial3            192.168.5.1     vrf_user                  up
```

Manually Configuring a BGP Router ID per VRF

Perform this task to manually configure a BGP router ID for each VRF. In this task, several address family configurations are shown and the router ID is configured in the IPv4 address family mode for one VRF instance. Step 22 shows you how to repeat certain steps to permit the configuration of more than one VRF on the same router.

Before you begin

This task assumes that you have previously created the VRF instances and associated them with interfaces. For more details, see the Configuring VRF Instances, on page 2 and the Associating VRF Instances with Interfaces, on page 4.

SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. **router bgp** *autonomous-system-number*
4. **no bgp default ipv4-unicast**
5. **bgp log-neighbor-changes**
6. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
7. **neighbor** {*ip-address*| *peer-group-name*} **update-source** *interface-type interface-number*
8. **address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]}
9. **neighbor** {*ip-address*| *peer-group-name*} **activate**
10. **neighbor** {*ip-address*| *peer-group-name*} **send-community**{**both**| **standard**| **extended**}
11. **exit-address-family**
12. **address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]}
13. **redistribute connected**
14. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
15. **neighbor** *ip-address* **local-as** *autonomous-system-number* [**no-prepend** [**replace-as** [**dual-as**]]]
16. **neighbor** {*ip-address*| *peer-group-name*} **ebgp-multihop**[*ttl*]
17. **neighbor** {*ip-address*| *peer-group-name*} **activate**
18. **neighbor** *ip-address* **allowas-in** [*number*]
19. **no auto-summary**
20. **no synchronization**
21. **bgp router-id** {*ip-address*| **auto-assign**}
22. Repeat Step 11 to Step 21 to configure another VRF instance.
23. **end**
24. **show ip bgp vpnv4** {**all**| **rd** *route-distinguisher*| **vrf** *vrf-name*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**no bgp default ipv4-unicast** **Example:**	Disables the IPv4 unicast address family for the BGP routing process.

	Command or Action	Purpose
	`Router(config-router)# no bgp default ipv4-unicast`	**Note** Routing information for the IPv4 unicast address family is advertised by default for each BGP routing session configured with the **neighbor remote-as** router configuration command unless you configure the **no bgp default ipv4-unicast**router configuration command before configuring the **neighbor remote-as** command. Existing neighbor configurations are not affected.
Step 5	**bgp log-neighbor-changes** **Example:** `Router(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.
Step 6	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.1 remote-as 45000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor. • In this example, the neighbor is an internal neighbor.
Step 7	**neighbor** {*ip-address*| *peer-group-name*} **update-source** *interface-type interface-number* **Example:** `Router(config-router)# neighbor 192.168.1.1 update-source loopback0`	Allows BGP sessions to use any operational interface for TCP connections. • In this example, BGP TCP connections for the specified neighbor are sourced with the IP address of the loopback interface rather than the best local address.
Step 8	**address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]} **Example:** `Router(config-router)# address-family vpnv4`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations. • The example creates a VPNv4 address family session.
Step 9	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 172.16.1.1 activate`	Activates the neighbor under the VPNv4 address family. • In this example, the neighbor 172.16.1.1 is activated.

	Command or Action	Purpose
Step 10	**neighbor** {*ip-address*| *peer-group-name*} **send-community**{**both**| **standard**| **extended**} **Example:** `Router(config-router-af)# neighbor 172.16.1.1 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor. • In this example, an extended communities attribute is sent to the neighbor at 172.16.1.1.
Step 11	**exit-address-family** **Example:** `Router(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 12	**address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]} **Example:** `Router(config-router)# address-family ipv4 vrf vrf_trans`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations. • The example specifies that the VRF instance named vrf_trans is to be associated with subsequent IPv4 address family configuration commands.
Step 13	**redistribute connected** **Example:** `Router(config-router-af)# redistribute connected`	Redistributes from one routing domain into another routing domain. • In this example, the **connected** keyword is used to represent routes that are established automatically when IP is enabled on an interface. • Only the syntax applicable to this step is displayed. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 14	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor 192.168.1.1 remote-as 40000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor. • In this example, the neighbor at 192.168.1.1 is an external neighbor.
Step 15	**neighbor** *ip-address* **local-as** *autonomous-system-number* [**no-prepend** [**replace-as** [**dual-as**]]]	Customizes the AS_PATH attribute for routes received from an eBGP neighbor.

	Command or Action	Purpose
	Example: `Router(config-router-af)# neighbor 192.168.1.1 local-as 50000 no-prepend`	• The autonomous system number from the local BGP routing process is prepended to all external routes by default. • Use the **no-prepend** keyword to not prepend the local autonomous system number to any routes received from the eBGP neighbor. • In this example, routes from the neighbor at 192.168.1.1 will not contain the local autonomous system number.
Step 16	**neighbor** {*ip-address*| *peer-group-name*} **ebgp-multihop**[*ttl*] **Example:** `Router(config-router-af)# neighbor 192.168.1.1 ebgp-multihop 2`	Accepts and attempts BGP connections to external peers residing on networks that are not directly connected. • In this example, BGP is configured to allow connections to or from neighbor 192.168.1.1, which resides on a network that is not directly connected.
Step 17	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 192.168.1.1 activate`	Activates the neighbor under the IPV4 address family. • In this example, the neighbor 192.168.1.1 is activated.
Step 18	**neighbor** *ip-address* **allowas-in** [*number*] **Example:** `Router(config-router-af)# neighbor 192.168.1.1 allowas-in 1`	Configures provider edge (PE) routers to allow the readvertisement of all prefixes that contain duplicate autonomous system numbers. • In the example, the PE router with autonomous system number 45000 is configured to allow prefixes from the VRF vrf-trans. The neighboring PE router with the IP address 192.168.1.1 is set to be readvertised once to other PE routers with the same autonomous system number.
Step 19	**no auto-summary** **Example:** `Router(config-router-af)# no auto-summary`	Disables automatic summarization and sends subprefix routing information across classful network boundaries.
Step 20	**no synchronization** **Example:** `Router(config-router-af)# no synchronization`	Enables the Cisco IOS XE software to advertise a network route without waiting for synchronization with an Internal Gateway Protocol (IGP).
Step 21	**bgp router-id** {*ip-address*| **auto-assign**} **Example:**	Configures a fixed router ID for the local BGP routing process.

	Command or Action	Purpose
	`Router(config-router-af)# bgp router-id 10.99.1.1`	• In this example, the specified BGP router ID is assigned for the VRF instance associated with this IPv4 address family configuration.
Step 22	Repeat Step 11 to Step 21 to configure another VRF instance.	--
Step 23	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.
Step 24	**show ip bgp vpnv4** {**all**| **rd** *route-distinguisher*| **vrf** *vrf-name*} **Example:** `Router# show ip bgp vpnv4 all`	(Optional) Displays VPN address information from the BGP table. • In this example, the complete VPNv4 database is displayed. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS Multiprotocol Label Switching Command Reference*.

Examples

The following sample output assumes that two VRF instances named vrf_trans and vrf_user were configured each with a separate router ID. The router ID is shown next to the VRF name.

```
Router# show ip bgp vpnv4 all
BGP table version is 5, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1:1 (default for vrf vrf_trans) VRF Router ID 10.99.1.2
*> 192.168.4.0      0.0.0.0                  0         32768 ?
Route Distinguisher: 42:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
*> 192.168.5.0      0.0.0.0                  0         32768 ?
```

Automatically Assigning a BGP Router ID per VRF

Perform this task to automatically assign a BGP router ID for each VRF. In this task, a loopback interface is associated with a VRF and the **bgp router-id** command is configured at the router configuration level to automatically assign a BGP router ID to all VRF instances. Step 9 shows you how to repeat certain steps to configure each VRF that is to be associated with an interface. Step 30 shows you how to configure more than one VRF on the same router.

Before you begin

This task assumes that you have previously created the VRF instances. For more details, see the Configuring VRF Instances, on page 2.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ip address** *ip-address mask* [**secondary**]
5. **exit**
6. **interface** *type number*
7. **ip vrf forwarding** *vrf-name* [**downstream** *vrf-name2*]
8. **ip address** *ip-address mask* [**secondary**]
9. Repeat Step 5 through Step 8 for each VRF to be associated with an interface.
10. **exit**
11. **router bgp** *autonomous-system-number*
12. **bgp router-id** {*ip-address*| **vrf auto-assign**}
13. **no bgp default ipv4-unicast**
14. **bgp log-neighbor-changes**
15. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
16. **neighbor** {*ip-address*| *peer-group-name*} **update-source** *interface-type interface-number*
17. **address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]}
18. **neighbor** {*ip-address*| *peer-group-name*} **activate**
19. **neighbor** {*ip-address*| *peer-group-name*} **send-community**{**both**| **standard**| **extended**}
20. **exit-address-family**
21. **address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]}
22. **redistribute connected**
23. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
24. **neighbor** *ip-address* **local-as** *autonomous-system-number* [**no-prepend** [**replace-as** [**dual-as**]]]
25. **neighbor** {*ip-address*| *peer-group-name*} **ebgp-multihop**[*ttl*]
26. **neighbor** {*ip-address*| *peer-group-name*} **activate**
27. **neighbor** *ip-address* **allowas-in** [*number*]
28. **no auto-summary**
29. **no synchronization**
30. Repeat Step 20 to Step 29 to configure another VRF instance.
31. **end**
32. **show ip bgp vpnv4** {**all**| **rd** *route-distinguisher*| **vrf** *vrf-name*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
	`Router> enable`	
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface loopback0`	Configures an interface type and enters interface configuration mode. • In this example, loopback interface 0 is configured.
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 172.16.1.1 255.255.255.255`	Configures an IP address. • In this example, the loopback interface is configured with an IP address of 172.16.1.1.
Step 5	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**interface** *type number* **Example:** `Router(config)# interface loopback1`	Configures an interface type and enters interface configuration mode. • In this example, loopback interface 1 is configured.
Step 7	**ip vrf forwarding** *vrf-name* [**downstream** *vrf-name2*] **Example:** `Router(config-if)# ip vrf forwarding vrf_trans`	Associates a VRF with an interface or subinterface. • In this example, the VRF named vrf_trans is associated with loopback interface 1. **Note** Executing this command on an interface removes the IP address. The IP address should be reconfigured.
Step 8	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 10.99.1.1 255.255.255.255`	Configures an IP address. • In this example, loopback interface 1 is configured with an IP address of 10.99.1.1.
Step 9	Repeat Step 5 through Step 8 for each VRF to be associated with an interface.	--
Step 10	**exit** **Example:**	Exits interface configuration mode and returns to global configuration mode.

	Command or Action	Purpose
	`Router(config-if)# exit`	
Step 11	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 12	**bgp router-id** {*ip-address*| **vrf auto-assign**} **Example:** `Router(config-router)# bgp router-id vrf auto-assign`	Configures a fixed router ID for the local BGP routing process. • In this example, a BGP router ID is automatically assigned for each VRF instance.
Step 13	**no bgp default ipv4-unicast** **Example:** `Router(config-router)# no bgp default ipv4-unicast`	Disables the IPv4 unicast address family for the BGP routing process. **Note** Routing information for the IPv4 unicast address family is advertised by default for each BGP routing session configured with the **neighbor remote-as** router configuration command unless you configure the **no bgp default ipv4-unicast**router configuration command before configuring the **neighbor remote-as** command. Existing neighbor configurations are not affected.
Step 14	**bgp log-neighbor-changes** **Example:** `Router(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.
Step 15	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.1 remote-as 45000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor. • In this example, the neighbor is an internal neighbor.
Step 16	**neighbor** {*ip-address*| *peer-group-name*} **update-source** *interface-type interface-number*	Allows BGP sessions to use any operational interface for TCP connections.

	Command or Action	Purpose
	Example: `Router(config-router)# neighbor 192.168.1.1 update-source loopback0`	• In this example, BGP TCP connections for the specified neighbor are sourced with the IP address of the loopback interface rather than the best local address.
Step 17	**address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]} **Example:** `Router(config-router)# address-family vpnv4`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations. • The example creates a VPNv4 address family session.
Step 18	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 172.16.1.1 activate`	Activates the neighbor under the VPNv4 address family. • In this example, the neighbor 172.16.1.1 is activated.
Step 19	**neighbor** {*ip-address*| *peer-group-name*} **send-community**{**both**| **standard**| **extended**} **Example:** `Router(config-router-af)# neighbor 172.16.1.1 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor. • In this example, an extended communities attribute is sent to the neighbor at 172.16.1.1.
Step 20	**exit-address-family** **Example:** `Router(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 21	**address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]} **Example:** `Router(config-router)# address-family ipv4 vrf vrf_trans`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations. • The example specifies that the VRF instance named vrf_trans is to be associated with subsequent IPv4 address family configuration mode commands.
Step 22	**redistribute connected** **Example:** `Router(config-router-af)# redistribute connected`	Redistributes from one routing domain into another routing domain. • In this example, the **connected** keyword is used to represent routes that are established automatically when IP is enabled on an interface. • Only the syntax applicable to this step is displayed. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

	Command or Action	Purpose
Step 23	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor 192.168.1.1 remote-as 40000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor. • In this example, the neighbor at 192.168.1.1 is an external neighbor.
Step 24	**neighbor** *ip-address* **local-as** *autonomous-system-number* [**no-prepend** [**replace-as** [**dual-as**]]] **Example:** `Router(config-router-af)# neighbor 192.168.1.1 local-as 50000 no-prepend`	Customizes the AS_PATH attribute for routes received from an eBGP neighbor. • The autonomous system number from the local BGP routing process is prepended to all external routes by default. • Use the **no-prepend** keyword to not prepend the local autonomous system number to any routes received from the eBGP neighbor. • In this example, routes from the neighbor at 192.168.1.1 will not contain the local autonomous system number.
Step 25	**neighbor** {*ip-address*| *peer-group-name*} **ebgp-multihop**[*ttl*] **Example:** `Router(config-router-af)# neighbor 192.168.1.1 ebgp-multihop 2`	Accepts and attempts BGP connections to external peers residing on networks that are not directly connected. • In this example, BGP is configured to allow connections to or from neighbor 192.168.1.1, which resides on a network that is not directly connected.
Step 26	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 192.168.1.1 activate`	Activates the neighbor under the IPV4 address family. • In this example, the neighbor 192.168.1.1 is activated.
Step 27	**neighbor** *ip-address* **allowas-in** [*number*] **Example:** `Router(config-router-af)# neighbor 192.168.1.1 allowas-in 1`	Configures provider edge (PE) routers to allow the readvertisement of all prefixes that contain duplicate autonomous system numbers. • In the example, the PE router with autonomous system number 45000 is configured to allow prefixes from the VRF vrf-trans. The neighboring PE router with

	Command or Action	Purpose
		the IP address 192.168.1.1 is set to be readvertised once to other PE routers with the same autonomous system number.
Step 28	**no auto-summary** **Example:** `Router(config-router-af)# no auto-summary`	Disables automatic summarization and sends subprefix routing information across classful network boundaries.
Step 29	**no synchronization** **Example:** `Router(config-router-af)# no synchronization`	Enables the Cisco IOS XE software to advertise a network route without waiting for synchronization with an Internal Gateway Protocol (IGP).
Step 30	Repeat Step 20 to Step 29 to configure another VRF instance.	--
Step 31	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.
Step 32	**show ip bgp vpnv4** {**all**| **rd** *route-distinguisher*| **vrf** *vrf-name*} **Example:** `Router# show ip bgp vpnv4 all`	(Optional) Displays VPN address information from the BGP table. • In this example, the complete VPNv4 database is displayed. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS Multiprotocol Label Switching Command Reference*.

Examples

The following sample output assumes that two VRF instances named vrf_trans and vrf_user were configured, each with a separate router ID. The router ID is shown next to the VRF name.

```
Router# show ip bgp vpnv4 all
BGP table version is 43, local router ID is 172.16.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1:1 (default for vrf vrf_trans) VRF Router ID 10.99.1.2
*> 172.22.0.0       0.0.0.0                  0         32768 ?
r> 172.23.0.0       172.23.1.1               0             0 3 1 ?
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
*> 10.52.1.0/24     172.23.1.1                             0 3 1 ?
*> 10.52.2.1/32     172.23.1.1                             0 3 1 3 i
```

```
*> 10.52.3.1/32     172.23.1.1                             0 3 1 3 i
*> 10.99.1.1/32     172.23.1.1               0             0 3 1 ?
*> 10.99.1.2/32     0.0.0.0                  0         32768 ?
Route Distinguisher: 10:1
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
Route Distinguisher: 42:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
r> 172.22.0.0       172.22.1.1               0             0 2 1 ?
*> 172.23.0.0       0.0.0.0                  0         32768 ?
*> 10.21.1.1/32     172.22.1.1                             0 2 1 2 i
*>i10.52.1.0/24     192.168.3.1              0    100      0 ?
*>i10.52.2.1/32     192.168.3.1              0    100      0 3 i
*>i10.52.3.1/32     192.168.3.1              0    100      0 3 i
*> 10.99.1.1/32     0.0.0.0                  0         32768 ?
*> 10.99.1.2/32     172.22.1.1               0             0 2 1 ?
```

Configuration Examples for Per-VRF Assignment of BGP Router ID

Manually Configuring a BGP Router ID per VRF Examples

The following example shows how to configure two VRFs--vrf_trans and vrf_user--with sessions between each other on the same router. The BGP router ID for each VRF is configured manually under separate IPv4 address families. The **show ip bgp vpnv4** command can be used to verify that the router IDs have been configured for each VRF. The configuration starts in global configuration mode.

```
ip vrf vrf_trans
 rd 45000:1
 route-target export 50000:50
 route-target import 40000:1
!
ip vrf vrf_user
 rd 65500:1
 route-target export 65500:1
 route-target import 65500:1
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
router bgp 45000
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 192.168.3.1 remote-as 45000
 neighbor 192.168.3.1 update-source Loopback0
 !
 address-family vpnv4
  neighbor 192.168.3.1 activate
  neighbor 192.168.3.1 send-community extended
  exit-address-family
 !
 address-family ipv4 vrf vrf_user
  redistribute connected
  neighbor 172.22.1.1 remote-as 40000
  neighbor 172.22.1.1 local-as 50000 no-prepend
  neighbor 172.22.1.1 ebgp-multihop 2
  neighbor 172.22.1.1 activate
  neighbor 172.22.1.1 allowas-in 1
```

```
  no auto-summary
  no synchronization
  bgp router-id 10.99.1.1
  exit-address-family
 !
 address-family ipv4 vrf vrf_trans
  redistribute connected
  neighbor 172.23.1.1 remote-as 50000
  neighbor 172.23.1.1 local-as 40000 no-prepend
  neighbor 172.23.1.1 ebgp-multihop 2
  neighbor 172.23.1.1 activate
  neighbor 172.23.1.1 allowas-in 1
  no auto-summary
  no synchronization
  bgp router-id 10.99.1.2
  exit-address-family
```

After the configuration, the output of the **show ip bgp vpnv4 all** command shows the router ID displayed next to the VRF name:

```
Router# show ip bgp vpnv4 all
BGP table version is 43, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 45000:1 (default for vrf vrf_trans) VRF Router ID 10.99.1.2
*> 172.22.0.0       0.0.0.0                  0         32768 ?
r> 172.23.0.0       172.23.1.1               0             0 3 1 ?
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
*> 10.52.1.0/24     172.23.1.1                             0 3 1 ?
*> 10.52.2.1/32     172.23.1.1                             0 3 1 3 i
*> 10.52.3.1/32     172.23.1.1                             0 3 1 3 i
*> 10.99.1.1/32     172.23.1.1               0             0 3 1 ?
*> 10.99.2.2/32     0.0.0.0                  0         32768 ?
Route Distinguisher: 50000:1
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
Route Distinguisher: 65500:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
r> 172.22.0.0       172.22.1.1               0             0 2 1 ?
*> 172.23.0.0       0.0.0.0                  0         32768 ?
*> 10.21.1.1/32     172.22.1.1                             0 2 1 2 i
*>i10.52.1.0/24     192.168.3.1              0    100      0 ?
*>i10.52.2.1/32     192.168.3.1              0    100      0 3 i
*>i10.52.3.1/32     192.168.3.1              0    100      0 3 i
*> 10.99.1.1/32     0.0.0.0                  0         32768 ?
*> 10.99.2.2/32     172.22.1.1               0             0 2 1 ?
```

The output of the **show ip bgp vpnv4 vrf** command for a specified VRF displays the router ID in the output header:

```
Router# show ip bgp vpnv4 vrf vrf_user
BGP table version is 43, local router ID is 10.99.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 65500:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
r> 172.22.0.0       172.22.1.1               0             0 2 1 ?
*> 172.23.0.0       0.0.0.0                  0         32768 ?
*> 10.21.1.1/32     172.22.1.1                             0 2 1 2 i
*>i10.52.1.0/24     192.168.3.1              0    100      0 ?
*>i10.52.2.1/32     192.168.3.1              0    100      0 3 i
*>i10.52.3.1/32     192.168.3.1              0    100      0 3 i
```

```
*> 10.99.1.1/32     0.0.0.0                  0         32768 ?
*> 10.99.2.2/32     172.22.1.1               0             0 2 1 ?
```

The output of the **show ip bgp vpnv4 vrf summary** command for a specified VRF displays the router ID in the first line of the output:

```
Router# show ip bgp vpnv4 vrf vrf_user summary
BGP router identifier 10.99.1.1, local AS number 45000
BGP table version is 43, main routing table version 43
8 network entries using 1128 bytes of memory
8 path entries using 544 bytes of memory
16/10 BGP path/bestpath attribute entries using 1856 bytes of memory
6 BGP AS-PATH entries using 144 bytes of memory
3 BGP extended community entries using 72 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 3744 total bytes of memory
BGP activity 17/0 prefixes, 17/0 paths, scan interval 15 secs
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
172.22.1.1      4     2      20      21       43    0    0 00:12:33        3
```

When the path is sourced in the VRF, the correct router ID is displayed in the output of the **show ip bgp vpnv4 vrf**command for a specified VRF and network address:

```
Router# show ip bgp vpnv4 vrf vrf_user 172.23.0.0
BGP routing table entry for 65500:1:172.23.0.0/8, version 22
Paths: (1 available, best #1, table vrf_user)
  Advertised to update-groups:
     2          3
  Local
    0.0.0.0 from 0.0.0.0 (10.99.1.1)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid, sourced, best
      Extended Community: RT:65500:1
```

Automatically Assigning a BGP Router ID per VRF Examples

The following three configuration examples show different methods of configuring BGP to automatically assign a separate router ID to each VRF instance:

Globally Automatically Assigned Router ID Using Loopback Interface IP Addresses Example

The following example shows how to configure two VRFs--vrf_trans and vrf_user--with sessions between each other on the same router. Under router configuration mode, BGP is globally configured to automatically assign each VRF a BGP router ID. Loopback interfaces are associated with individual VRFs to source an IP address for the router ID. The **show ip bgp vpnv4** command can be used to verify that the router IDs have been configured for each VRF.

```
ip vrf vrf_trans
 rd 45000:1
 route-target export 50000:50
 route-target import 40000:1
!
ip vrf vrf_user
 rd 65500:1
 route-target export 65500:1
 route-target import 65500:1
!
interface Loopback0
```

```
 ip address 10.1.1.1 255.255.255.255
!
interface Loopback1
 ip vrf forwarding vrf_user
 ip address 10.99.1.1 255.255.255.255
!
interface Loopback2
 ip vrf forwarding vrf_trans
 ip address 10.99.2.2 255.255.255.255
!
router bgp 45000
 bgp router-id vrf auto-assign
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 192.168.3.1 remote-as 45000
 neighbor 192.168.3.1 update-source Loopback0
 !
address-family vpnv4
  neighbor 192.168.3.1 activate
  neighbor 192.168.3.1 send-community extended
  exit-address-family
 !
 address-family ipv4 vrf vrf_user
  redistribute connected
  neighbor 172.22.1.1 remote-as 40000
  neighbor 172.22.1.1 local-as 50000 no-prepend
  neighbor 172.22.1.1 ebgp-multihop 2
  neighbor 172.22.1.1 activate
  neighbor 172.22.1.1 allowas-in 1
  no auto-summary
  no synchronization
  exit-address-family
 !
 address-family ipv4 vrf vrf_trans
  redistribute connected
  neighbor 172.23.1.1 remote-as 50000
  neighbor 172.23.1.1 local-as 2 no-prepend
  neighbor 172.23.1.1 ebgp-multihop 2
  neighbor 172.23.1.1 activate
  neighbor 172.23.1.1 allowas-in 1
  no auto-summary
  no synchronization
  exit-address-family
```

After the configuration, the output of the **show ip bgp vpnv4 all** command shows the router ID displayed next to the VRF name. Note that the router IDs used in this example are sourced from the IP addresses configured for loopback interface 1 and loopback interface 2. The router IDs are the same as in the Manually Configuring a BGP Router ID per VRF Examples, on page 18.

```
Router# show ip bgp vpnv4 all
BGP table version is 43, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 45000:1 (default for vrf vrf_trans) VRF Router ID 10.99.2.2
*> 172.22.0.0       0.0.0.0                  0         32768 ?
r> 172.23.0.0       172.23.1.1               0             0 3 1 ?
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
*> 10.52.1.0/24     172.23.1.1                             0 3 1 ?
*> 10.52.2.1/32     172.23.1.1                             0 3 1 3 i
*> 10.52.3.1/32     172.23.1.1                             0 3 1 3 i
*> 10.99.1.1/32     172.23.1.1               0             0 3 1 ?
```

```
*> 10.99.1.2/32     0.0.0.0                  0         32768 ?
Route Distinguisher: 50000:1
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
Route Distinguisher: 65500:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
r> 172.22.0.0       172.22.1.1               0             0 2 1 ?
*> 172.23.0.0       0.0.0.0                  0         32768 ?
*> 10.21.1.1/32     172.22.1.1                             0 2 1 2 i
*>i10.52.1.0/24     192.168.3.1              0    100      0 ?
*>i10.52.2.1/32     192.168.3.1              0    100      0 3 i
*>i10.52.3.1/32     192.168.3.1              0    100      0 3 i
*> 10.99.1.1/32     0.0.0.0                  0         32768 ?
*> 10.99.1.2/32     172.22.1.1               0             0 2 1 ?
```

Globally Automatically Assigned Router ID with No Default Router ID Example

The following example shows how to configure a router and associate a VRF that is automatically assigned a BGP router ID when no default router ID is allocated.

```
ip vrf vpn1
 rd 45000:1
 route-target export 45000:1
 route-target import 45000:1
!
interface Loopback0
 ip vrf forwarding vpn1
 ip address 10.1.1.1 255.255.255.255
!
router bgp 45000
 bgp router-id vrf auto-assign
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 !
 address-family ipv4 vrf vpn1
  neighbor 172.22.1.2 remote-as 40000
  neighbor 172.22.1.2 activate
  no auto-summary
  no synchronization
  exit-address-family
```

Assuming that a second router is configured to establish a session between the two routers, the output of the **show ip interface brief** command shows only the VRF interfaces that are configured.

```
Router# show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
Serial2/0/0            unassigned      YES NVRAM  administratively down down
Serial3/0/0            unassigned      YES NVRAM  administratively down down
Loopback0              10.1.1.1        YES NVRAM  up                    up
```

The **show ip vrf** command can be used to verify that a router ID is assigned for the VRF:

```
Router# show ip vrf
Name                             Default RD          Interfaces
  vpn1                             45000:1           Loopback0
VRF session is established:
```

Per-VRF Automatically Assigned Router ID Example

The following example shows how to configure two VRFs--vrf_trans and vrf_user--with sessions between each other on the same router. Under the IPv4 address family associated with an individual VRF, BGP is configured to automatically assign a BGP router ID. Loopback interfaces are associated with individual VRFs to source an IP address for the router ID. The output of the **show ip bgp vpnv4** command can be used to verify that the router IDs have been configured for each VRF.

```
ip vrf vrf_trans
 rd 45000:1
 route-target export 50000:50
 route-target import 40000:1
!
ip vrf vrf_user
 rd 65500:1
 route-target export 65500:1
 route-target import 65500:1
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
interface Loopback1
 ip vrf forwarding vrf_user
 ip address 10.99.1.1 255.255.255.255
!
interface Loopback2
 ip vrf forwarding vrf_trans
 ip address 10.99.2.2 255.255.255.255
!
router bgp 45000
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 192.168.3.1 remote-as 45000
 neighbor 192.168.3.1 update-source Loopback0
 !
address-family vpnv4
  neighbor 192.168.3.1 activate
  neighbor 192.168.3.1 send-community extended
  exit-address-family
!
 address-family ipv4 vrf vrf_user
  redistribute connected
  neighbor 172.22.1.1 remote-as 40000
  neighbor 172.22.1.1 local-as 50000 no-prepend
  neighbor 172.22.1.1 ebgp-multihop 2
  neighbor 172.22.1.1 activate
  neighbor 172.22.1.1 allowas-in 1
  no auto-summary
  no synchronization
  bgp router-id auto-assign
  exit-address-family
 !
 address-family ipv4 vrf vrf_trans
  redistribute connected
  neighbor 172.23.1.1 remote-as 50000
  neighbor 172.23.1.1 local-as 40000 no-prepend
  neighbor 172.23.1.1 ebgp-multihop 2
  neighbor 172.23.1.1 activate
  neighbor 172.23.1.1 allowas-in 1
  no auto-summary
  no synchronization
  bgp router-id auto-assign
  exit-address-family
```

After the configuration, the output of the **show ip bgp vpnv4 all** command shows the router ID displayed next to the VRF name. Note that the router IDs used in this example are sourced from the IP addresses configured for loopback interface 1 and loopback interface 2.

```
Router# show ip bgp vpnv4 all
BGP table version is 43, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 45000:1 (default for vrf vrf_trans) VRF Router ID 10.99.2.2
*> 172.22.0.0       0.0.0.0                  0         32768 ?
r> 172.23.0.0       172.23.1.1               0             0 3 1 ?
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
*> 10.52.1.0/24     172.23.1.1                             0 3 1 ?
*> 10.52.2.1/32     172.23.1.1                             0 3 1 3 i
*> 10.52.3.1/32     172.23.1.1                             0 3 1 3 i
*> 10.99.1.1/32     172.23.1.1               0             0 3 1 ?
*> 10.99.1.2/32     0.0.0.0                  0         32768 ?
Route Distinguisher: 50000:1
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
Route Distinguisher: 65500:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
r> 172.22.0.0       172.22.1.1               0             0 2 1 ?
*> 172.23.0.0       0.0.0.0                  0         32768 ?
*> 10.21.1.1/32     172.22.1.1                             0 2 1 2 i
*>i10.52.1.0/24     192.168.3.1              0    100      0 ?
*>i10.52.2.1/32     192.168.3.1              0    100      0 3 i
*>i10.52.3.1/32     192.168.3.1              0    100      0 3 i
*> 10.99.1.1/32     0.0.0.0                  0         32768 ?
*> 10.99.1.2/32     172.22.1.1               0             0 2 1 ?
```

Additional References

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS IP Routing: BGP Command Reference*
MPLS commands: complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Multiprotocol Label Switching Command Reference*
Cisco IOS master command list, all releases	Cisco IOS Master Command List, All Releases

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for Per-VRF Assignment of BGP Router ID

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

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

Table 1: Feature Information for Per-VRF Assignment of BGP Router ID

Feature Name	Releases	Feature Information
Per-VRF Assignment of BGP Router ID	Cisco IOS XE Release 2.1	The Per-VRF Assignment of BGP Router ID feature introduces the ability to have VRF-to-VRF peering in Border Gateway Protocol (BGP) on the same router. BGP is designed to refuse a session with itself because of the router ID check. The per-VRF assignment feature allows a separate router ID per VRF using a new keyword in the existing **bgp router-id** command. The router ID can be manually configured for each VRF or can be assigned automatically either globally under address family configuration mode or for each VRF. This feature was introduced on the Cisco ASR 1000 Series Aggregation Services Routers. The following commands were introduced or modified by this feature: **bgp router-id**, **show ip bgp vpnv4**.