BGP Neighbor and Session Configuration

This chapter explains how to configure BGP neighbors to ensure reliable and scalable routing in your network. You'll learn how to simplify management with neighbor groups, fine-tune session parameters, and apply powerful policy controls using BGP communities.

BGP neighbor groups

A BGP neighbor group is a configuration template that:

- stores address family-independent and address family-dependent settings
- enables multiple BGP neighbors to inherit shared configuration parameters, and
- allows centralized management and consistent application of BGP neighbor policies.

Configuration inheritance

After you configure a neighbor group, each BGP neighbor can inherit its configuration by using the **use** command. By default, a neighbor configured to use a neighbor group inherits all configuration from that group, including address family-independent and address family-dependent settings.

Overriding inherited configuration

You can override inherited settings by directly configuring commands for the neighbor. Alternatively, associate session groups or address family groups with the neighbor using the **use** command

Configuring address family-independent parameters

Configure address family-independent parameters directly within the neighbor group configuration mode.

Configuring address family-dependent parameters

To configure address family-dependent parameters, enter the address family submode under the neighbor group by using the **address-family** command.

Assigning options to a neighbor group

After specifying the neighbor group name with the **neighbor group** command, assign the configuration options you want the group to use. Downstream neighbors inherit these options.

Suppose you have several BGP neighbors that require the same password and update-source configuration. Instead of configuring these parameters for each neighbor individually, define them in a neighbor group and attach the group to each neighbor with the **use** command.

Configure a BGP neighbor group

Centralize and streamline BGP configuration by creating a BGP neighbor group and applying its settings to BGP neighbors.

BGP neighbor groups let you manage common settings, such as address family, remote AS number, and route policies across multiple BGP neighbors. By grouping neighbors with similar configurations, you can simplify management and reduce configuration errors.

Before you begin

- Ensure you have administrative access to the command-line interface of the router.
- Verify that the router supports BGP and is properly licensed.
- Identify the autonomous system (AS) numbers and IP addresses for both local and remote peers.
- Prepare any required route policies to apply to the neighbor group.

Follow these steps to configure a BGP neighbor group and apply it to a neighbor:

Procedure

Step 1 Enter BGP configuration mode and specify the local AS number and address family (such as IPv4 or IPv6 unicast).

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
```

Step 2 Configure a neighbor group by specifying a remote autonomous system (AS) number for that group. Apply the address family, such as IPv4 or IPv6, to the BGP neighbor group. Additionally, you can apply inbound or outbound route policies to that neighbor group.

Example:

```
Router(config-bgp)# neighbor-group nbr-grp-A
Router(config-bgp-nbrgrp)# remote-as 2002
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
Router(config-bgp-nbrgrp-af)# route-policy drop-as-1234 in
Router(config-bgp-nbrgrp-af)# exit
Router(config-bgp-nbrgrp)# exit
```

Step 3 Configure a BGP neighbor to inherit configuration from the specified neighbor group.

Example:

```
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbr)# use neighbor-group nbr-grp-A
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# commit
```

The BGP neighbor group is configured, and its settings are successfully applied to the specified neighbor. Any changes to the group are consistently applied to all assigned neighbors for centralized management.

Example

This example illustrates centralized configuration using BGP neighbor groups and route policies:

```
route-policy pass-all
  pass
end-policy
router bgp 109
  address-family ipv4 unicast
     network 172.16.0.0 255.255.0.0
     network 192.16831.7.0 255.255.0.0
     neighbor 172.16.200.1
       remote-as 167
       exit
  address-family ipv4 unicast
     route-policy pass-all in
     route-policy pass-out out
     neighbor 172.26.234.2
       remote-as 109
       exit
  address-family ipv4 unicast
     neighbor 172.26.64.19
       remote-as 99
       exit
  address-family ipv4 unicast
      route-policy pass-all in
      route-policy pass-all out
```

In this example:

- The BGP router is configured to belong to AS 109.
- The router advertises two networks, 172.16.0.0 and 192.168.7.0, as originating from AS 109.
- There are three neighbor routers, two external and one internal:
  - The first and third neighbors use different AS numbers.
  - The second neighbor shares AS 109 and is identified as an internal neighbor.
  - Route policies are applied to inbound and outbound routing updates.

This setup allows the router to share routing information about the two networks with all configured neighbors, while distinguishing between internal and external BGP peers based on the AS number.

What to do next

- Verify the BGP neighbor and group configuration using **show** commands (for example, **show running-config** or **show bgp neighbors**) to confirm the correct application of settings.

- Monitor BGP peer status and logs to ensure neighbor establishment and exchange of routing information.

Adjust BGP timers

Set or modify the keepalive and hold timers for BGP neighbors to control the frequency of keepalive messages and determine when a neighbor is declared down.

BGP uses timers to manage keepalive messages and neighbor status. You can set global timer defaults and override them for individual neighbors if needed.

Before you begin

- Ensure you are in privileged EXEC mode on the router.
- Know the autonomous system (AS) number for your BGP process.
- Have the neighbor IP addresses ready.

Procedure

Enter BGP configuration mode, and set default BGP keepalive and hold timers for all neighbors or a specific BGP neighbor.

The values set using the **timers bgp** command in router configuration mode apply globally, but you can override them on a per-neighbor basis using the **timers** command in neighbor configuration mode.

Example:

```
Router# configure
Router(config)# router bgp 123
Router(config-bgp)# timers bgp 30 90
Router(config-bgp)# neighbor 172.16.40.24
Router(config-bgp-nbr)# timers 60 220
```

Soft reconfiguration

A soft reconfiguration is a BGP feature that

- stores inbound routing updates received from a neighbor
- enables policy changes to be applied without clearing the BGP session, and
- provides the ability to reapply or review routing policies on previously received routes.

How the soft-reconfiguration inbound command works

When the **soft-reconfiguration inbound** command is configured for a BGP neighbor, the router stores the original, unmodified BGP paths received from that neighbor. If the neighbor supports route refresh, a route

refresh request can be sent to retrieve routing information. If route refresh is not supported, soft reconfiguration ensures that all received routes are available locally for policy reapplication.

Conditions for storing BGP routes during soft reconfiguration

- Storing updates from a neighbor is possible only if either the neighbor supports route refresh or the **soft-reconfiguration inbound** command is configured.
- The **soft-reconfiguration inbound always** command forces the router to store a copy of the received routes even if route refresh is supported by the neighbor.

Memory considerations for soft reconfiguration

Soft reconfiguration is memory intensive, as all received paths must be stored until they are no longer needed.

Suppose a BGP router has received multiple routing updates from a neighbor. If network administrators need to change routing policies or filters, soft reconfiguration allows them to reapply new policy rules to previously stored routes without resetting the BGP session or losing any routing updates.

Configure BGP soft reconfiguration for a neighbor

Enable BGP to store incoming route updates from a neighbor, allowing policy changes to be applied with a soft clear instead of resetting the BGP session.

Use soft reconfiguration when a BGP neighbor does not support route refresh, or when you require the flexibility to apply new policies and retrieve original routes without a full session reset.

Before you begin

- Identify the autonomous system (AS) number.
- Obtain the IP address of the BGP neighbor.
- Ensure sufficient memory is available, as soft reconfiguration can be resource-intensive.

Procedure

Enter BGP configuration mode, and enable soft reconfiguration for the neighbor.

Example:

```
Router(config)# router bgp 120
Router(config-bgp)# neighbor 172.16.40.24
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# soft-reconfiguration inbound always
```

BGP large communities

BGP large community is a BGP routing attribute that

- enables grouping of network destinations for routing policies
- encodes both 4-byte autonomous system numbers (ASNs) and operator-assigned local values, and
- supports complex route-matching and policy operations using community values.

BGP communities allow network operators to tag routes with information that can influence route acceptance, rejection, preference, or redistribution. Traditional BGP community attributes use 4 bytes, making them insufficient for encoding both 4-byte ASNs (introduced in RFC 6793) and local values. BGP extended communities can encode 4-byte ASNs in the global administrator field, but limitations in the local administrator field still exist.

Importance of BGP large communities

To address these limitations, BGP large communities were introduced. Each large community consists of three 4-byte, unsigned integers separated by colons (for example, 1:2:3). The first field typically encodes the ASN, while the other two fields are set by the operator.

Policy matching with BGP large communities

BGP large communities can be matched or set in route-policy languages (RPL) using flexible syntax and are compatible with various expressions for advanced policy matching.

Expressions used in BGP large community policies

You can use these expressions in route-policy statements to match or set BGP large community values (fields are separated by colons):

- `[x..y]` — Specifies a range between x and y, inclusive.
- `*` — Stands for any number.
- `peeras` — Substitutes the ASN of the BGP neighbor (inbound or outbound as appropriate).
- `not-peeras` — Matches any number except the ASN of the neighbor.
- `private-as` — Matches a private ASN in the ranges [64512..65534] and [4200000000..4294967294].

Regular expressions can also be used for matches or deletes:

- **ios-regex example:** `'^5:.*:7$'` is equivalent to the expression `5:*:7`.

Restrictions and guidelines for BGP large communities

These restrictions and guidelines apply to BGP large communities.

- All functionality available in the BGP community attribute is available for the BGP large-community attribute.
- The **send-community-ebgp** command is required for the BGP speaker to send large communities to eBGP neighbors.

- There are no well-known large communities defined.
- The `peeras` expression cannot be used in a large-community-set.
- The `peeras` expression can only be used in large-community match or delete statements that appear in route policies at neighbor-in or neighbor-out attach points.
- The `not-peeras` expression cannot be used in a large-community-set or in policy set statements.

Configure a named large-community set

Create a named set of BGP large communities for use in route-policy matching and set statements.

Before you begin

Ensure you are in router configuration mode on a Cisco IOS XR device.

Procedure

Create a large-community set by specifying its name. Add large community values, each in the format `A:B:C` or with expressions as needed.

Example:

```
Router(config)# large-community-set catbert
Router(config-largecomm)#  1: 2: 3,
Router(config-largecomm)#  peeras:2:3
Router(config-largecomm)# end-set
```

Match BGP large communities using route policies

Configure policies to match routes based on presence or pattern of large communities.

You can match if any, all, or a subset of a route's large communities correspond to specific criteria.

Procedure

Step 1 Enter route-policy configuration mode, and use the **if large-community matches-any** command to match any element of a large community set.

Example:

```
Router(config)# route-policy elbonia
Router(config-rpl)# if large-community matches-any (1:2:3, 4:5:*) then
Router(config-rpl)# set local-preference 94
Router(config-rpl)# endif
Router(config-rpl)# end-policy
```

Step 2 Use the **if large-community matches-every** command to match every specification.

Example:

```
Router(config)# route-policy bob
Router(config-rpl)# if large-community matches-every (*:*:3, 4:5:*) then
Router(config-rpl)# set local-preference 94
Router(config-rpl)# endif
Router(config-rpl)# end-policy
```

Step 3 Use the **if large-community matches-within** command to match within a large community set. This command is similar to the **large-community matches-any** command but every large community in the route must match at least one match specification. If the route has no large communities, the condition matches.

Example:

```
Router(config)# route-policy bob
Router(config-rpl)# if large-community matches-within (*:*:3, 4:5:*) then
Router(config-rpl)# set local-preference 103
Router(config-rpl)# endif
Router(config-rpl)# end-policy
```

Set BGP large community attributes in a route policy

Assign BGP large community attributes to routes within a policy for use in route filtering and redistribution.

Before you begin

Ensure you have an existing large-community set or inline set, and are in route-policy configuration mode.

Procedure

Step 1 Use the **set large-community** command, specifying a set name or inline values.

Example:

```
Router(config)# route-policy mordac
Router(config-rpl)# set large-community (1:2:3, peeras:2:3)
Router(config-rpl)# end-set
```

Step 2 (Optional) Include the **additive** keyword to retain existing large-community values.

Example:

```
router(config-rpl)# set large-community catbert additive
```

The **additive** keyword keeps the existing large communities on a route and adds any new large communities you specify. It does not create duplicate entries.

Filter routes without large communities

Configure a route-policy that matches routes missing the large-community attribute.

Procedure

Enter route-policy configuration mode, and use the **if large-community is-empty** command to filter routes without large communities.

Example:

```
Router(config)# route-policy lrg_comm_rp4
Router(config-rpl)# if large-community is-empty then
Router(config-rpl)# set local-preference 104
Router(config-rpl)# endif
Router(config-rpl)# end-policy
```

Apply attribute filtering for BGP large communities

Filter BGP update messages based on large-community attributes using an attribute-filter group.

Before you begin

Ensure you are in BGP router configuration mode.

Procedure

Create an attribute-filter group specifying the LARGE-COMMUNITY attribute and desired action (for example, discard).

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# attribute-filter group dogbert
Router(config-bgp-attrfg)# attribute LARGE-COMMUNITY discard
Router(config-bgp-attrfg)# neighbor 10.0.1.101
Router(config-bgp-nbr)# remote-as 6461
Router(config-bgp-nbr)# update in filtering
Router(config-nbr-upd-filter)# attribute-filter group dogbert
```

Updates containing specified large-community attributes from the neighbor are discarded as configured.

Delete BGP large communities from route policies

Remove specific BGP large-communities from routes using route-policy configuration.

Before you begin

Ensure you are in route-policy configuration mode.

Procedure

Use the **delete large-community** command with specific matching criteria, such as regular expressions, 'all', or inline specifications to delete large communities.

Example:

```
Router(config)# route-policy lrg_comm_rp2
Router(config-rpl)# delete large-community in (ios-regex '^100000:')
Router(config-rpl)# delete large-community all
Router(config-rpl)# delete large-community not in (peeras:*:*, 41289:*:*)
```

The specified large communities are removed from the affected routes according to the deletion criteria used.

Show commands for BGP large communities

- To display routes containing specified large communities, use the **show bgp large-community** *community-list* **exact-match** command.
  - If **exact-match** keyword is specified, only routes with the exact set of listed communities are shown. Otherwise, routes with additional large communities are included.

  Example:

```
Router# show bgp large-community 1:2:3 5:6:7
Thu Mar 23 14:40:33.597 PDT
BGP router identifier 10.4.4.4, local AS number 3
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 66
BGP main routing table version 66
BGP NSR Initial initsync version 3 (Reached)
BGP NSR/ISSU Sync-Group versions 66/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
* 10.0.0.3/32         10.10.10.3               0     94      0 ?
* 10.0.0.5/32         10.11.11.5               0             0 5 ?
```

- To display the large community for a specific network, use the **show bgp** *ip-address* / *prefix-length* command. The output displays route entries and the large-community attributes attached.

  Example:

```
Router# show bgp 10.3.3.3/32
Thu Mar 23 14:36:15.301 PDT
BGP routing table entry for 10.3.3.3/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 42          42
Last Modified: Mar 22 20:04:46.000 for 18:31:30
```

```
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    10.11.11.5
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    10.11.11.5
  Local
    10.10.10.3 from 10.10.10.3 (10.3.3.3)
      Origin incomplete, metric 0, localpref 94, valid, internal, best, group-best
      Received Path ID 0, Local Path ID 0, version 42
      Community: 258:259 260:261 262:263 264:265
      Large Community: 1:2:3 5:6:7 4123456789:4123456780:4123456788
```

Per-VRF label allocation for VPN routes

A per-VRF label allocation for VPN routes is a label assignment method that

- assigns a single label per VPN routing and forwarding (VRF) instance rather than per individual route prefix
- conserves label resources, which is especially important on low-end platforms with limited label capacity, and
- enables more efficient advertisement and management of imported VPN routes across different route distinguishers (RDs).

Table 1: Feature history table

Feature Name	Release Information	Feature Description
Per-VRF label allocation for VPN routes	Release 25.3.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) * This feature is now supported on Cisco 8404-SYS-D routers.
Per-VRF label allocation for VPN routes	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]). This feature is now supported on Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
Per-VRF label allocation for VPN routes	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC:Q200, P100], 8700 [ASIC: P100, K100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) This feature modifies the default label allocation from per-prefix to per-VRF by allowing you to enforce per-VRF label allocation for imported VPN routes using the **advertise vpn-imported label-mode per-vrf** command. This feature introduces these changes: **CLI:** • **advertise vpn-imported label-mode per-vrf** **YANG Data Model:** • `Cisco-IOS-XR-um-vrf-cfg.yang` (see GitHub, YANG Data Models Navigator)

Default label allocation behavior

When a Route Reflector (RR) that is also a Provider Edge (PE) router is configured with the same route distinguisher (RD) and the next-hop-self option, the system uses per-prefix mode for local label allocation by default. This default applies even if Prefix Independent Convergence (PIC) is enabled. In this scenario, remote VPN routes with the same RD and matching route targets also receive per-prefix labels.

Label exhaustion in low-end platforms

On devices with limited label capacity, using per-prefix label allocation for imported VPN routes can exhaust available labels. Switching to per-VRF label allocation conserves label space and prevents label exhaustion on these platforms.

Modified label allocation behavior

This feature enables per-VRF label allocation for imported VPN routes with the same RD. By using the **advertise vpn-imported label-mode per-vrf** command, you can override the default per-prefix allocation in favor of per-VRF label assignment:

Support for differing RDs

For routes with different RDs, the default behavior assigns per-prefix labels to routes with remote RDs, and these routes are advertised. Imported VPN routes are not advertised by default. When per-VRF label allocation is enabled:

- Routes with remote RDs do not receive labels and are not advertised.
- Imported VPN routes are assigned a single per-VRF label and are advertised.

Per-VRF label allocation example

If you enable per-VRF label allocation on a low-end platform that previously exhausted label space under per-prefix allocation, imported VPN routes will now share a single label per VRF, resolving label exhaustion and optimizing label usage.

Limitations of Per-VRF label allocation for VPN routes

Ensure you understand and adhere to the following limitations when applying per-VRF label allocation for VPN routes:

- Apply per-VRF label allocation only to VPNv4 and VPNv6 routes.
- Use per-VRF label allocation only with VRF-imported prefixes.
- Configure per-VRF label allocation only when the next-hop is changed. If the next-hop is not changed, the label mode defaults to per-prefix even if the per-VRF configuration is present.
- Do not use per-VRF label allocation with EVPN.
- Use the **is-imported-path** keyword for import match only at the neighbor outbound route-policy attach-point.

Configure Per-VRF label allocation for VPN routes

You can configure the Per-VRF label allocation for VPN routes feature in two scenarios:

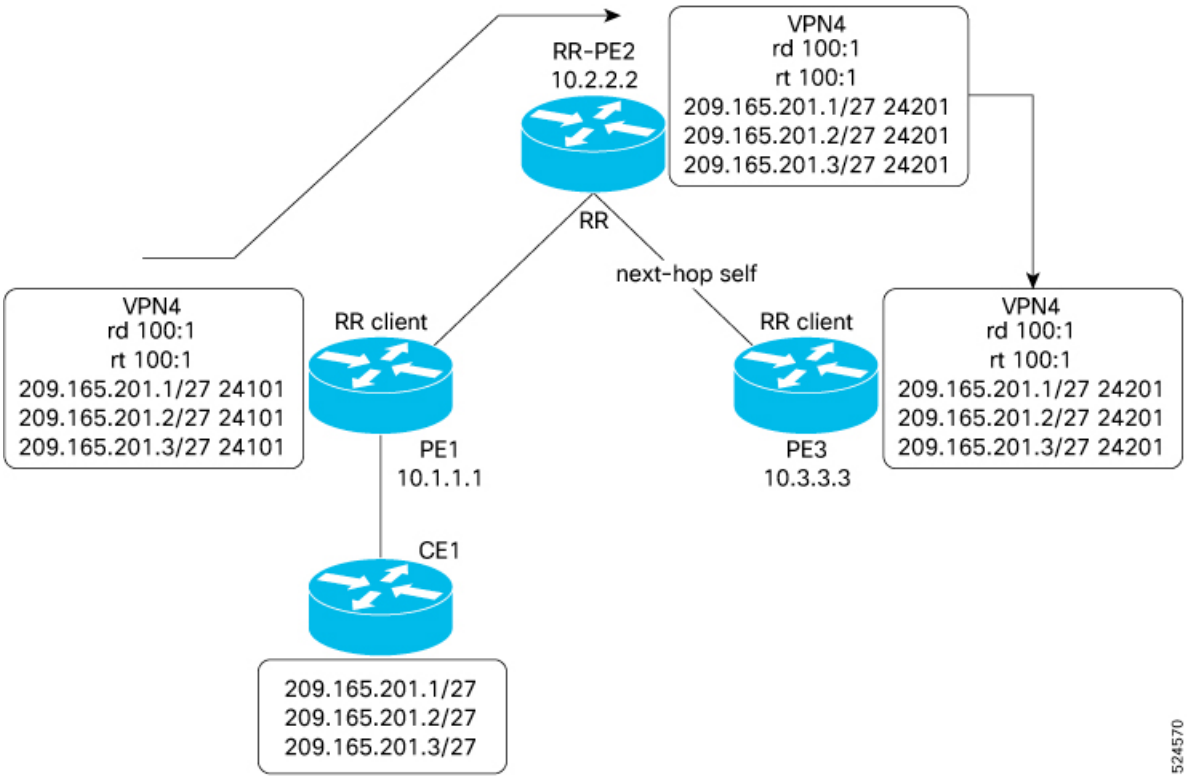
- Scenario 1: RR and PE routers are configured with the same RD
- Scenario 2: RR and PE routers are configured with different RDs

Configure Per-VRF label allocation for VPN routes in same RD scenarios

Enable per-VRF label allocation for VPN routes when Route Reflectors (RR) and Provider Edge (PE) routers are configured with the same Route Distinguisher (RD).

This is a sample topology where the RR and PEs are configured with the same RD.

Figure 1: Sample topology for same RD configuration



Follow these steps to configure per-VRF label allocation for VPN routes in the same RD scenario:

Procedure

Step 1 Enable per-VRF label allocation on the RR-PE (for example, RR-PE2).

Example:

```
Router# configure
Router(config)# vrf vrf_1
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# advertise vpn-imported label-mode per-vrf
```

Step 2 Configure the BGP neighbor with the next-hop-self attribute.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 10.3.3.3
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# next-hop-self
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# address-family ipv6 unicast
Router(config-bgp-nbr-af)# next-hop-self
Router(config-bgp-nbr-af)# exit
```

Note
You can configure **next-hop-self** directly, as shown above, or set it within a neighbor outbound route-policy.

Step 3 Use these commands to verify that the per-VRF label allocation is enabled.

Example:

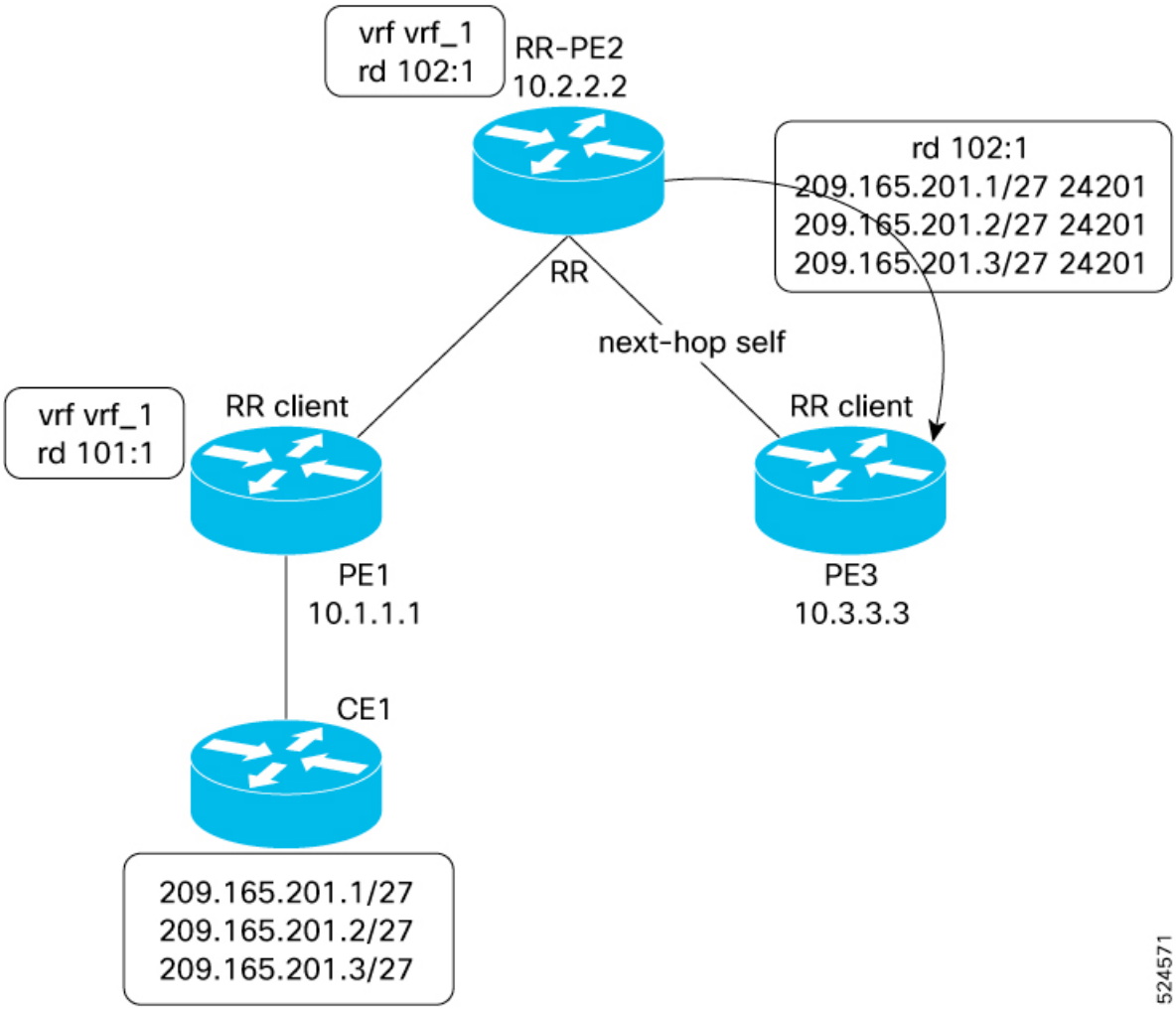
```
Router# show bgp label table
Router# show bgp vpnv4 unicast rd
Router# show mpls label table
Router# show controllers npu resources
```

Configure per-VRF label allocation for VPN routes with different RDs

Enable per-VRF label allocation for VPN routes in scenarios where the route reflector (RR) and provider edge (PE) routers use different route distinguishers (RDs).

This is a sample topology where the RR and PEs are configured with different RDs.

Figure 2: Sample topology for different RD configuration



Before you begin

- Ensure you have access to the relevant routers and their configuration interfaces.
- Confirm BGP sessions are established between RRs and PEs.

Procedure

Step 1 Enable per-VRF label allocation on the RR-PE (for example, RR-PE2).

Example:

```
Router# configure
Router(config)# vrf vrf_1
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# advertise vpn-imported label-mode per-vrf
```

Step 2 Configure the BGP neighbor with the next-hop-self attribute for IPv4 and IPv6 address families.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 10.3.3.3
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# next-hop-self
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# address-family ipv6 unicast
Router(config-bgp-nbr-af)# next-hop-self
Router(config-bgp-nbr-af)# exit
```

Step 3 Create and apply route policies to allow only imported VPN paths and local paths, blocking remote VPN paths.

Example:

```
Router(config)# route-policy rp-advertise-imported
Router(config-rpl)#  if destination is-imported-path or source in (0.0.0.0) then
Router(config-rpl-if)# pass
Router(config-rpl-if)# else
Router(config-rpl-else)# drop
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
```

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 10.3.3.3
Router(config-bgp-nbr)# address-family vpnv4 unicast
Router(config-bgp-nbr-af)# route-policy rp-advertise-imported out
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# address-family vpnv6 unicast
Router(config-bgp-nbr-af)# route-policy rp-advertise-imported out
Router(config-bgp-nbr-af)# exit
```

Step 4 Attach the **no-advertise** community in the neighbor inbound policy to prevent the advertisement of remote RD paths.

Example:

```
Router(config)# community-set com-set-no-advertise
Router(config-comm)# no-advertise
Router(config-comm)# end-set
```

```
Router(config)# route-policy rpl-set-no-advertise
Router(config-rpl)#  set community com-set-no-advertise
Router(config-rpl)# end-policy
```

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 10.1.1.1
Router(config-bgp-nbr)# address-family vpnv4 unicast
Router(config-bgp-nbr-af)# route-policy rpl-set-no-advertise in
Router(config-bgp-nbr-af)# exit
```

Step 5 Remove the **no-advertise** community from the imported paths to enable advertisement of imported VPN paths.

Example:

```
Router(config)# route-policy no-set-community
Router(config-rpl)# delete community in com-set-no-advertise
Router(config-rpl)# end-policy
```

```
Router# configure
Router(config)# vrf vrf_1
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# import route-policy no-set-community
```

Step 6 Use these commands to verify that the per-VRF label allocation is enabled.

Example:

```
Router# show bgp label table
Router# show bgp vpnv4 unicast rd
Router# show mpls label table
Router# show controllers npu resources
```