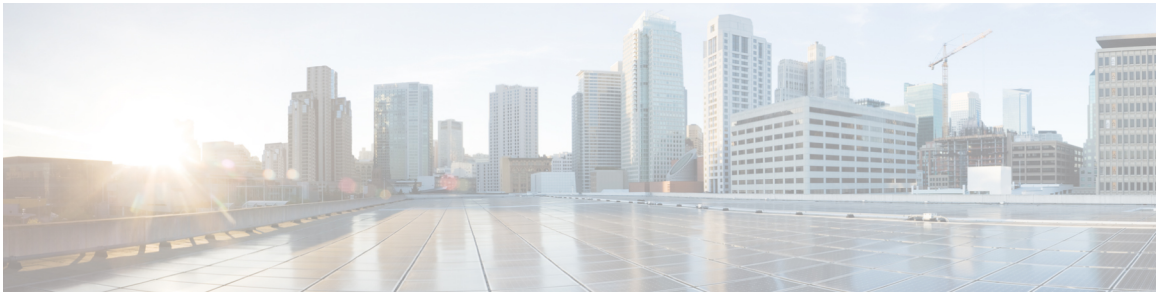
BGP Configuration Guide for Cisco 8000 Series Routers, Cisco IOS XR Releases

First Published: 2025-11-14

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883



CONTENTS

Preface

This preface contains these sections:

Preface

This cumulative guide provides a single, continuously updated version that includes all the latest IOS XR features and release updates. It simplifies your experience by letting you bookmark one link and access the complete guide, instead of navigating through multiple release-specific versions.

Specific changes or updates tied to individual releases are clearly called out within the relevant sections. For a list of features introduced in a specific release, refer to the Release Notes or the IOS XR Feature Finder.

The table lists the release numbers for which this document has been updated since its initial publication.

Table 1: Changes to this document

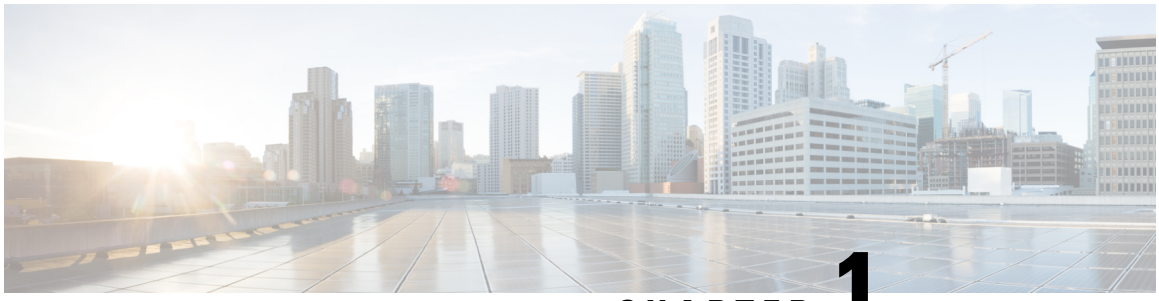
Date	Summary
November 2025	First published.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business results you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco DevNet.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.



CHAPTER 1

YANG Data Models for BGP Features

This chapter provides information about the YANG data models for BGP features.

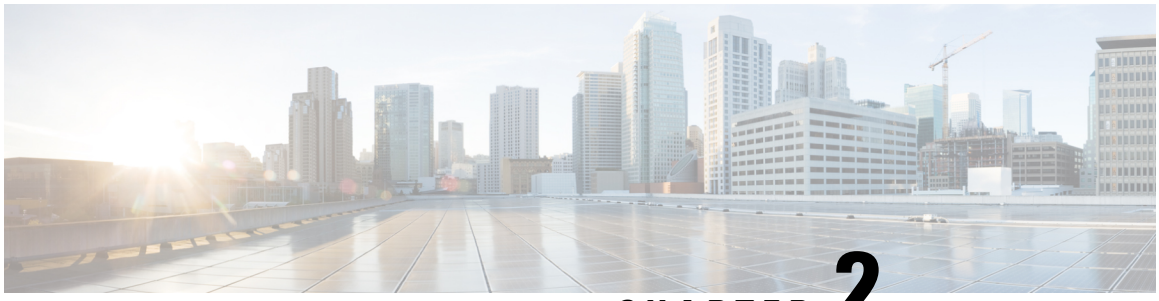
Using YANG Data Models

Cisco IOS XR supports a programmatic way of configuring and collecting operational data of a network device using YANG data models. Although configurations using CLIs are easier and human-readable, automating the configuration using model-driven programmability results in scalability.

The data models are available in the release image, and are also published in the Github repository. Navigate to the release folder of interest to view the list of supported data models and their definitions. Each data model defines a complete and cohesive model, or augments an existing data model with additional XPaths. To view a comprehensive list of the data models supported in a release, navigate to the *Available-Content.md* file in the repository.

You can also view the data model definitions using the YANG Data Models Navigator tool. This GUI-based and easy-to-use tool helps you explore the nuances of the data model and view the dependencies between various containers in the model. You can view the list of models supported across Cisco IOS XR releases and platforms, locate a specific model, view the containers and their respective lists, leaves, and leaf lists presented visually in a tree structure. This visual tree form helps you get insights into nodes that can help you automate your network.

To get started with using the data models, see the *Programmability Configuration Guide*.



CHAPTER 2

Service Layer API for BGP Features

This chapter provides information about the service layer API for BGP features.

Service Layer API

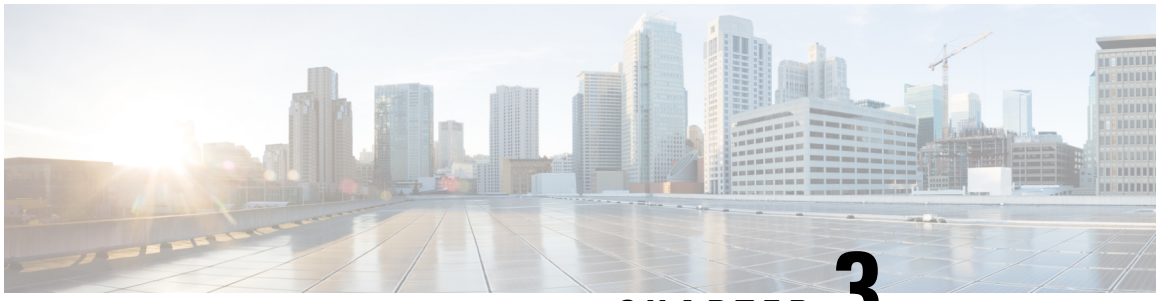
The Service Layer API is a model-driven API that leverages Google's remote procedure call (gRPC) to provide a high-performance interface, offers direct access to the routing infrastructure layer, provides flexibility to integrate custom protocols, agents, or controllers while IOS XR manages low-level tasks like resource conflict resolution, route preference, transactional notifications, and data plane abstractions, supports integration of applications, routing protocols, and controllers in languages such as C++, Python, and Go without requiring additional packages, and enables clients to subscribe to required state event notifications, facilitating real-time monitoring and management of network devices.

Activating the Service Layer API server on your router enables it to process API requests. You must also develop a Service Layer API client to interact with the server for retrieving data, configuring settings, or managing the network. See the *Telemetry Configuration Guide for Cisco 8000 Series Routers* for details on activating the Service Layer API server and developing the Service Layer API client.

Creating the Service Layer API provides access to system resources, enabling external applications to interact with and configure the router programmatically. See Cisco IOS-XR Service Layer for details on creating the Service Layer API.

Table 2: Feature History Table

Feature name	Release Information	Description
Service Layer API for BGP	Release 25.3.1	The Service Layer API enhances the system's performance by directly accessing the routing infrastructure layer using gRPC. It also allows flexible integration of custom protocols. For detailed information on Service Layer API for BGP, see Cisco IOS XR SL API documentation



CHAPTER 3

Introduction to BGP and Deployment Prerequisites

This chapter introduces the key concepts of BGP, explains how it differs from other routing protocols, and outlines the technical and operational prerequisites needed for a successful BGP implementation.

Border Gateway Protocol

Border Gateway Protocol (BGP) is an exterior gateway protocol that

- enables loop-free interdomain routing between autonomous systems
- exchanges routing information between different networks operated by distinct organizations, and
- supports policy-based routing decisions for scalable and reliable Internet operation.

Autonomous system

An autonomous system (AS) is a group of routers managed under a single technical administration that operates as a unified routing entity. Each AS consists of one or more IP networks under the control of a single organization and is assigned a unique Autonomous System Number (ASN).

Comparison of routing protocol types

To understand BGP, it's essential to distinguish between two types of routing protocols:

- **Interior Gateway Protocols (IGPs):** These protocols such as OSPF, IS-IS, and EIGRP, are used within a single AS. They determine the best paths for routing data within the internal network of that AS.
- **Exterior Gateway Protocols (EGPs):** These protocols, primarily BGP, are used for routing between different ASes. They enable communication and the exchange of routing information across different networks.

Feature	Exterior Gateway Protocols (EGPs)	Interior Gateway Protocols (IGPs)
Scope	Inter-AS (between organizations)	Intra-AS (within an organization)
Typical protocols	BGP	OSPF, IS-IS, EIGRP, RIP
Routing policy support	Policy-based (highly flexible)	Generally least-cost or shortest path
Scalability	Highly scalable for large networks	Suitable for limited, internal domains

- An Internet Service Provider uses BGP to exchange routing information with other service providers, ensuring that data can travel efficiently between different networks on the global Internet.
- Enterprises with multi-homed Internet connections deploy BGP to manage traffic policies and redundancy across their upstream providers.

How BGP works

Summary

The key components involved in BGP operation include:

- BGP: A protocol that uses TCP as its transport protocol to exchange routing information between routers.
- Peer routers (BGP peers/neighbors): Two routers that establish a TCP connection to exchange BGP routing information.
- BGP routing table: A table that contains the routing information exchanged between peers.
- Keepalive packets: Packets that are periodically exchanged between peers to verify that the connection is still active.

BGP operates by establishing a TCP connection between routers (peers) to exchange network reachability information, which includes paths represented by Autonomous System Numbers (ASNs). This information is used to construct a loop-free routing graph, enabling efficient routing and policy enforcement. Once connected, peers exchange complete routing tables initially, followed by incremental updates for changes such as new routes, modifications, or withdrawals. To maintain the connection, periodic keepalive packets are exchanged, and notification packets are sent for errors or special conditions, ensuring reliable inter-domain routing.

Workflow

These stages describe how BGP works.

1. Establishing a TCP connection:
   - Two BGP routers, referred to as BGP peers or neighbors, establish a TCP connection.
   - These routers exchange messages to open the connection and confirm the connection parameters.
2. Exchanging network reachability information:

- After the connection is established, BGP routers exchange network reachability information.
- This information includes the complete paths represented by BGP autonomous system numbers (ASNs) that a route should take to reach a destination network.
- The exchanged information helps construct a graph of autonomous systems. This graph shows loop-free paths and identifies where routing policies can be applied to enforce restrictions on routing behavior.

3. Constructing a loop-free routing graph:
   - BGP uses the exchanged network reachability information to construct a graph.
   - This graph identifies loop-free autonomous systems and highlights where routing policies can be applied to enforce restrictions on routing behavior.
4. Exchanging routing tables:

   When the BGP connection is established, the peers exchange their complete BGP routing tables.
5. Sending incremental updates:

   After the initial exchange, BGP routers send only incremental updates as changes occur in the routing table. These updates may include new routes, modified routes, or route withdrawals.
6. Tracking routing table versioning:

   BGP keeps a version number for the routing table. This version number is identical across all BGP peers and is updated whenever changes are made to the table due to routing information updates.
7. Maintaining the connection:
   - BGP routers send periodic **keepalive** packets between peers to ensure that the connection remains active.
   - BGP routers send **notification** packets when errors or special conditions occur.

Result

This process ensures that BGP operates efficiently, maintains reliable communication, and adapts to changes in routing information dynamically.

BGP router identifier

Each BGP router needs a unique router identifier (ID) to establish BGP sessions. This ID is sent to BGP peers in the OPEN message when a BGP session starts.

Methods for determining BGP router ID

BGP tries to obtain a router ID in this order of preference:

1. **Configured router ID**: BGP uses the router ID set explicitly with the **bgp router-id** command in router configuration mode.

2. **Highest IPv4 address on loopback interface**: If no router ID is configured, BGP uses the highest IPv4 address from a loopback interface, provided the router was booted with a saved loopback address configuration.
3. **Primary IPv4 address of the first configured loopback**: If no loopback address exists in the saved configuration, BGP uses the primary IPv4 address of the first loopback interface that is configured.

What happens if BGP fails to obtain a router ID

If none of these methods for obtaining a router ID succeeds, BGP does not have a router ID and cannot establish any peering sessions with BGP neighbors.

In such an instance:

- The system logs an error message.
- The **show bgp summary** command displays a router ID of 0.0.0.0.

Behavior after a router ID is assigned

Once BGP obtains a router ID, it uses that ID continuously, even if a better option becomes available based on the rules. This behavior prevents unnecessary flapping of BGP sessions.

Exception

If the current router ID becomes invalid, for instance due to a down interface or configuration change, BGP selects a new one according to the previously outlined rules. However, this change resets all established peering sessions.

Best practice

We strongly recommend configuring the router ID using the **bgp router-id** command. This helps prevent unexpected changes to the router ID and avoids potential BGP session flapping.

Prerequisites for implementing BGP

User group and task ID requirements

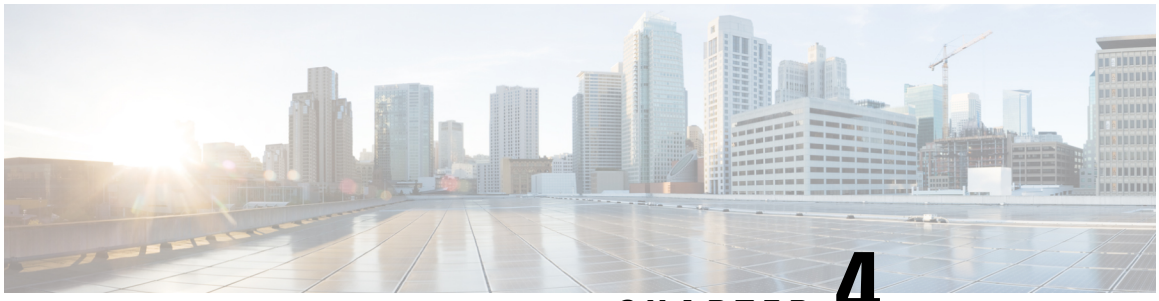
- You must be part of a user group associated with a task group that includes the required task IDs.
- Refer to the command reference guides to find the task IDs necessary for each command.
- Contact your AAA administrator for assistance if you suspect that your user group assignment is restricting access to use a command.

BGP table memory requirements

- The current Internet BGP table contains approximately 1.1 million IPv4 routes and 200,000 IPv6 routes. Ensure your router has at least 5.5 GB of RAM to manage the full Internet BGP table, with an average of two paths per route.
- You must plan for higher memory requirements as the IPv6 Internet routing table continues to expand.

Hardware requirements

Use the Service Edge (SE) version of Route Processor (RP) cards, Route Switch Processor (RSP) cards, or fixed chassis routers for devices that need to maintain a full BGP table.



CHAPTER 4

Enabling and Setting up BGP

This chapter provides clear, step-by-step guidance for enabling and configuring BGP on your network devices. It covers essential setup tasks, best practices for managing BGP neighbors and route policies, and introduces multi-instance and multi-AS features to help you scale and optimize your network.

Enable BGP routing

Enable BGP routing and establish BGP neighbor relationships to control route exchange in your network.

Configure BGP on your router to participate in inter-domain routing. You must define route policies and associate address families with each neighbor to control which routes are exchanged. This is essential for both internal BGP (iBGP) and eBGP peering.

Before you begin

- Ensure your BGP router has a router identifier, such as a configured loopback address.
- Configure at least one address family within the BGP router configuration, and explicitly configure the same address family under each BGP neighbor configuration.
- For an external BGP (eBGP) neighbor:
  - Configure both inbound and outbound route policies on the neighbor using the**route-policy** command, or
  - Allow route exchange for all eBGP neighbors by using the **bgp unsafe-ebgp-policy** command.
- If peering with eBGP neighbors that are not directly connected:
  - Note that, by default, BGP checks for direct connectivity and does not establish a connection if peers are not directly connected.
  - Use the **ignore-connected-check** command to enable peering between loopback interfaces of non-directly connected routers. Set the TTL value to 1 when using this command.
  - Configure the **egp-multihop** command for eBGP peers separated by intermediate routers, setting the TTL value to at least the number of hops between peers (for example, use TTL 3 for two intermediate hops).

Procedure

Step 1 Define a route policy to control routing behavior.

Example:

```
Router# configure
Router(config)# route-policy drop-as-1234
Router(config-rpl)# if as-path passes-through '1234' then
Router(config-rpl)# apply check-communities
Router(config-rpl)# else
Router(config-rpl)# pass
Router(config-rpl)# endif
Router(config-rpl)# end-policy
```

Step 2 Enter BGP configuration mode. Configure the router ID, and specify the address family to define the type of routes BGP will handle, such as IPv4 or IPv6 unicast.

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# bgp router-id 192.168.70.24
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
```

Step 3 Configure a BGP neighbor with an IP address, assign a remote AS number, and associate an address family. Optionally, configure route policies to control which routes are exchanged.

Example:

```
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# address-family ipv4 unicast
Routing(config-bgp-nbr-af)# route-policy drop-as-1234 in
Routing(config-bgp-nbr-af)# commit
```

BGP routing is enabled. The router can now establish BGP neighbor relationships and exchange routing information as configured.

Example

Sample configuration to enable BGP:

```
route-policy pass-all
  pass
end-policy

route-policy set_next_hop_agg_v4
  set next-hop 10.0.0.1
end-policy

route-policy set_next_hop_static_v4
  if (destination in static) then
    set next-hop 10.1.0.1
  else
    drop
```

```
  endif
end-policy

route-policy set_next_hop_agg_v6
  set next-hop 2001:DB8::121
end-policy

route-policy set_next_hop_static_v6
  if (destination in static) then
    set next-hop 2001:DB8::201 else
  drop
  endif
end-policy

router bgp 65000
bgp fast-external-fallover disable
bgp confederation peers 65001 65002
bgp confederation identifier 1
bgp router-id 10.1.1.1
address-family ipv4 unicast
  aggregate-address 10.2.0.0/24 route-policy
set_next_hop_agg_v4
  aggregate-address 10.3.0.0/24
  redistribute static route-policy
set_next_hop_static_v4
  address-family ipv6 unicast
  aggregate-address 2001:DB8:2::/32 route-policy
set_next_hop_agg_v6
  aggregate-address 2001:DB8:3::/32
  redistribute static route-policy
set_next_hop_static_v6
  neighbor 10.0.101.60
    remote-as 65000
    address-family ipv4 unicast
  neighbor 10.0.101.61
    remote-as 65000
    address-family ipv4 unicast
  neighbor 10.0.101.62
    remote-as 3
    address-family ipv4 unicast
      route-policy pass-all in
      route-policy pass-all out
  neighbor 10.0.101.64
    remote-as 5
    update-source Loopback0
    address-family ipv4 unicast
      route-policy pass-all in
      route-policy pass-all out
```

What to do next

Verify BGP neighbor relationships and route exchange using the **show bgp summary** command.

BGP multi-instance and multi-AS support

BGP multi-instance and multi-AS are mechanisms that

- combine services from multiple routers into a single IOS XR router

- achieve address family (AF) isolation by configuring different address families in separate BGP instances, and
- increase session scale by distributing peering sessions across multiple BGP instances.

This feature enables each BGP instance on a router to operate with a unique Autonomous System (AS) number. This provides enhanced flexibility and scalability for complex network environments.

Benefits of BGP multi-instance and multi-AS

These are the benefits of the BGP multi-instance and multi-AS feature:

- **Simplified network management**: You can consolidate services from multiple routers into a single IOS XR router. This reduces overall complexity and decreases the effort needed to manage the network.
- **Enhanced control and isolation**: By isolating different address families (AFs) in separate BGP instances, you gain better traffic management and policy enforcement.
- **Scalability for large networks**: By distributing the load across multiple BGP instances, this feature supports a higher number of peering sessions and larger prefix tables.
- **Improved network stability**: By providing faster and more reliable BGP convergence in certain scenarios, this feature increases your network uptime and responsiveness.
- **High availability**: With full support for BGP features, including Nonstop Routing (NSR), your network runs continuously and remains resilient.

Requirements for configuring BGP multi-instance and multi-AS

Follow these requirements when configuring BGP multi-instance and multi-autonomous system (multi-AS) features:

Instance limits and identification

- Limit the router to a maximum of 4 BGP instances.
- Assign a unique router ID to each BGP instance.

Address family configuration

- Configure only one address family under each BGP instance. VPNv4, VPNv6, and RT-Constrain address families can be configured under multiple BGP instances.
- Place IPv4 or IPv6 unicast within the same BGP instance where you configure IPv4 or IPv6 labeled-unicast.
- Place IPv4 or IPv6 multicast within the same BGP instance where you configure IPv4 or IPv6 unicast.

Configuration management

- Commit all configuration changes for one BGP instance at the same time.
- Commit configuration changes for only one BGP instance at a time.

Operational recommendations

Use a unique BGP update-source in the default VRF for each BGP instance when you peer with the same remote router.

Configure multiple BGP instances for a specific autonomous system

Configure several BGP instances within a single autonomous system, assigning a unique router ID to each instance.

Before you begin

- Ensure you have administrative access to the router.
- Collect autonomous system (AS) numbers and router ID values you want to use.

Follow these steps to configure multiple BGP instances in a specific autonomous system:

Procedure

Step 1 Enter BGP instance configuration mode for the specific autonomous system and specify a unique name for each instance. Assign a unique router ID value to the BGP instance.

Example:

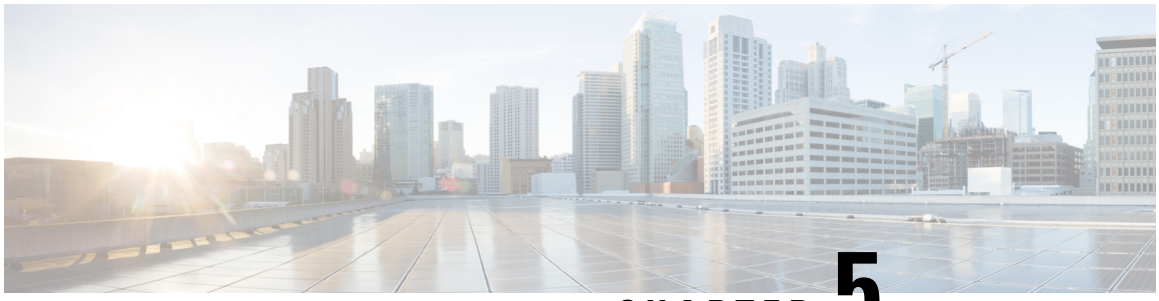
```
Router# configure
Router(config)# router bgp 100 instance inst1
Router(config-bgp)# bgp router-id 10.0.0.0
Router(config-bgp)# commit
```

Ensure that the router ID you assign is unique for each BGP instance in the autonomous system.

Restriction
Commit all configuration changes for each BGP instance separately. You cannot commit changes for multiple instances at the same time.

Step 2 Repeat step 1 for each additional BGP instance you want to configure within the same autonomous system, specifying a unique instance name and router ID each time.

Each BGP instance in the specified autonomous system has a unique router ID and its configuration is committed individually.



CHAPTER 5

Core BGP Configurations

This chapter provides a comprehensive guide to configuring Border Gateway Protocol (BGP) on Cisco IOS XR devices. You will learn fundamental configuration methods, how to use templates for scalable deployments, and how inheritance and precedence rules affect BGP operations. The chapter also covers address family behavior and advanced features such as BGP confederations. By following this chapter, you will be ready to deploy and manage BGP in a variety of network scenarios.

Key aspects of BGP configurations

Cisco IOS XR software uses a neighbor-based model for BGP configuration. All BGP settings are managed at the individual neighbor level.

Key aspects of BGP configuration in Cisco IOS XR include:

- **Neighbor-specific configuration:** All settings for a particular neighbor must be defined under the neighbor configuration. You must apply configuration statements to each BGP neighbor individually.
- **Configuration and update groups:** IOS XR BGP does not support traditional peer groups. Instead, it uses configuration groups and update groups to simplify management and improve performance.
  - **Configuration groups** allow you to create templates with reusable configuration statements that can be applied to multiple neighbors. This approach reduces configuration errors and ensures consistency across multiple BGP neighbor configurations.
  - **Update groups** are automatically generated by the system based on common outbound policies among BGP neighbors. Update groups enable efficient sharing of update messages among neighbors that have the same outbound policies, improving scalability and network performance.

BGP configuration modes

BGP configurations are organized into multiple command modes, with each mode enabling you to configure specific aspects of BGP operation. You can enter each mode at the CLI prompt, and within that mode, use the **?** command for a list of available configuration commands.

These are common BGP configuration modes with examples demonstrating how to access each:

Mode	Description	Example
Global BGP configuration	Applies to overall BGP settings for the router	`router bgp 65000` sets BGP process for AS 65000
Address-family ipv4 unicast	Configures IPv4 unicast-specific routing parameters	`address-family ipv4 unicast` enables IPv4 unicast routes
Address-family ipv4 multicast	Configures IPv4 multicast-specific routing	`address-family ipv4 multicast` enables multicast routing
Address-family ipv6 unicast	Configures IPv6 unicast-specific routing parameters	`address-family ipv6 unicast` enables IPv6 unicast routes
Address-family vpnv4	Configures routing for VPNv4 addresses (L3VPN)	`address-family vpnv4` enables MPLS VPNv4 routing
Peer-group configuration	Sets attributes for a group of BGP neighbors	`neighbor PEERS peer-group` defines a peer group
Individual neighbor configuration	Customizes settings for a single neighbor	`neighbor 192.0.2.1 remote-as 65001`

Mode	Description	Typical Use Case
Global BGP configuration	Applies to overall BGP settings for the router	Set AS number, router ID, and global behaviors
Address-family ipv4 unicast	Configures IPv4 unicast-specific routing parameters	Control IPv4 unicast routing and policies
Address-family ipv4 multicast	Configures IPv4 multicast-specific routing	Manage IPv4 multicast routing in BGP
Address-family ipv6 unicast	Configures IPv6 unicast-specific routing parameters	Enable BGP for IPv6 unicast operations
Address-family vpnv4	Configures routing for VPNv4 addresses (L3VPN)	Implement MPLS VPNs
Peer-group configuration	Sets attributes for a group of BGP neighbors	Simplifies management of multiple neighbors
Individual neighbor configuration	Customizes settings for a single neighbor	Fine-tune parameters per neighbor

Router configuration mode

Use this mode to configure global BGP settings for an autonomous system.

```
Router# configuration
Router(config)# router bgp 140
Router(config-bgp)#
```

Router address family configuration mode

Use this mode to configure address family–specific BGP settings.

```
Router(config)# router bgp 112
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)#
```

Neighbor configuration mode

Use this mode to configure settings for a specific BGP neighbor.

```
Router(config)# router bgp 140
Router(config-bgp)# neighbor 10.0.0.1
Router(config-bgp-nbr)#
```

VRF configuration mode

Use this mode to configure BGP for a specific Virtual Routing and Forwarding (VRF) instance.

```
Router(config)# router bgp 140
Router(config-bgp)# vrf vrf_A
Router(config-bgp-vrf)#
```

VRF neighbor configuration mode

Use this mode to configure a BGP neighbor within a specific VRF.

```
Router(config)# router bgp 140
Router(config-bgp)# vrf vrf_A
Router(config-bgp-vrf)# neighbor 11.0.1.2
Router(config-bgp-vrf-nbr)#
```

VRF neighbor address family configuration mode

Use this mode to configure address family settings for a BGP neighbor in a VRF.

```
RP/0/RP0/CPU0:router(config)# router bgp 112
RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_A
RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 10.0.1.2
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)#
```

VPNv6 address family configuration mode

Use this mode to configure VPNv6-specific BGP address family settings.

```
Router(config)# router bgp 150
Router(config-bgp)# address-family vpnv6 unicast
Router(config-bgp-af)#
```

L2VPN address family configuration mode

Use this mode to configure Layer 2 VPN-specific BGP address family settings.

```
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn vpls-vpws
Router(config-bgp-af)#
```

BGP submodes for neighbor configuration

BGP neighbor configuration in Cisco IOS XR uses submodes to simplify and organize neighbor and address-family specific settings. Each submode determines the command scope and reduces repetitive command syntax for efficient configuration.

Neighbor submode

Neighbor submode allows you to configure attributes for a specific BGP neighbor without prefixing each command with the **neighbor** keyword.

You can enter neighbor submode by using the **neighbor** *ip-address* command in BGP configuration mode.

Benefit:

Streamlines configuration by targeting a single neighbor, reducing command repetition.

Example:

```
Router(config-bgp)# neighbor 192.23.1.2
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# address-family ipv4 unicast
```

Address-family submode

Address-family submode, accessed within neighbor submode, enables you to configure address-family-specific settings, such as IPv4 or IPv6 unicast, for a selected neighbor.

You can enter address-family submode for a specific neighbor by using the **address-family <afi> <safi>** command within neighbor submode.

Benefit:

Groups configuration by address family, minimizes command repetition, and clarifies parameter application.

Example:

```
Router(config-bgp)# neighbor 2002::2
Router(config-bgp-nbr)# remote-as 2023
Router(config-bgp-nbr)# address-family ipv6 unicast
Router(config-bgp-nbr-af)# next-hop-self
Router(config-bgp-nbr-af)# route-policy one in
```

Note

- Commands entered in a submode automatically apply to the appropriate neighbor and address-family context.
- Leaving a submode returns you to the higher-level mode, and changes only apply within the current submode context.

BGP configuration templates

BGP configuration templates in Cisco IOS XR ensure consistent, scalable, and efficient BGP configuration, especially in large-scale deployments. Templates allow you to define common configurations once and apply them to multiple BGP neighbors, reducing configuration errors, and simplifying ongoing maintenance.

Types of BGP configuration templates

There are three primary types of BGP configuration templates, each designed for specific configuration requirements:

- **af-group:** This template type is used to define address family–specific settings for groups of neighbors. It allows you to configure commands that are dependent on the address family, such as IPv4 or IPv6. Neighbors can inherit these settings by referencing the af-group with the **use** command.
- **session-group:** The session-group template is intended for address family–independent (global) settings, which apply to groups of neighbors regardless of address family. Typical commands set at this level include global BGP options. Neighbors inherit these configuration commands from the session-group template by using the **use** command.
- **neighbor-group:** This template brings together both address family–independent and address family–dependent configurations. It enables you to apply a comprehensive set of commands, both global and specific to a group of neighbors. Neighbors assigned to a neighbor-group inherit all its configuration settings using the **use** command. Additionally, a neighbor-group can reference both af-group and session-group templates, allowing for hierarchical inheritance.

Inheritance behavior

- When you assign a neighbor to a group using the **use** command, the neighbor inherits all settings from the group unless you explicitly override a setting at the neighbor level.
- **neighbor-group** templates can include references to both session-groups and af-groups, allowing hierarchical inheritance.
- If a setting is configured directly under a neighbor, it overrides inherited values from any template.
- Some inherited configuration settings may be hidden if multiple group types are layered.

Key effects and attributes

- Commands entered at the session-group level define global, address family–independent options. These match the commands available in **neighbor** submode.
- Commands entered at the af-group level define address family–specific options, as found in the **neighbor-address-family** configuration submode.
- Commands at the neighbor-group level include both global and address family–dependent options. These commands may reference both af-group and session-group templates through the **use** command.

Configuration examples

af-group:

```
Router(config)# router bgp 140
Router(config-bgp)# af-group afmcast1 address-family ipv4 unicast
Router(config-bgp-afgrp)#
```

session-group:

```
Router# router bgp 140
Router(config-bgp)# session-group session1
Router(config-bgp-sngrp)#
```

neighbor-group:

```
Router(config)# router bgp 123
Router(config-bgp)# neighbor-group nbrgroup1
Router(config-bgp-nbrgrp)#
```

neighbor-group with address family:

```
Router(config)# router bgp 140
Router(config-bgp)# neighbor-group nbrgroup1
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
Router(config-bgp-nbrgrp-af)#
```

Note Directly configured settings always take precedence over inherited template settings.

Precedence of BGP configuration template inheritance

When multiple configuration templates are applied to a BGP neighbor, the final settings are determined based on a specific order of precedence. Understanding this order is essential to troubleshooting and verifying BGP behavior.

Order of inheritance precedence

When multiple configuration sources apply to a BGP neighbor or group, the effective value is determined in the following order (from highest to lowest precedence):

1. **Direct neighbor configuration:** Values set directly on the neighbor override all inherited values.
2. **Session group or address family group:** If not set directly, values from a session group or an address family group override those from a neighbor group.
3. **Neighbor group:** If not set in the neighbor or higher-level groups, values from a neighbor group (including any values it inherits) are applied.
4. **System default:** If a value is not configured at any level, the Cisco IOS XR system default is used.

Inheritance rules summary

- Neighbors can inherit configuration from both session groups and neighbor groups, as well as address family groups.
- Neighbor groups can inherit from session groups, address family groups, and other neighbor groups.
- Session groups can inherit from other session groups.
- Address family groups can inherit from other address family groups.

Address family inheritance

- Inheritance rules apply to both address family-independent and address family-dependent configurations.
- If a property can be set in both an address family-dependent and independent context, the most specific value takes precedence, following the same order.

Examples

Example 1: Direct neighbor configuration takes highest precedence

If you set `advertisement-interval` both on a neighbor group and directly on a neighbor, the neighbor setting is applied.

```
Router(config)# router bgp 140
Router(config-bgp)# neighbor-group AS_1
Router(config-bgp-nbrgrp)# advertisement-interval 15
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# neighbor 10.1.1.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# use neighbor-group AS_1
Router(config-bgp-nbr)# advertisement-interval 20
```

This sample output from the **show bgp neighbors** command shows that the advertisement interval used is 20 seconds:

```
Router# show bgp neighbors 10.1.1.1

BGP neighbor is 10.1.1.1, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 20 seconds

 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.1
  eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%

  Connections established 0; dropped 0
  Last reset 00:00:14, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

Example 2: Session-group or address family group overrides neighbor group

If a value is set both in a session-group (or address family group) and in a neighbor group, but not directly on the neighbor, the session group or address family group value is used.

```
Router(config)# router bgp 140
Router(config-bgp)# session-group AS_2
Router(config-bgp-sngrp)# advertisement-interval 15
Router(config-bgp-sngrp)# exit
Router(config-bgp)# neighbor-group AS_1
Router(config-bgp-nbrgrp)# advertisement-interval 20
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# neighbor 192.168.0.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# use session-group AS_2
Router(config-bgp-nbr)# use neighbor-group AS_1
```

This sample output from the **show bgp neighbors** command shows that the advertisement interval used is 15 seconds:

```
Router# show bgp neighbors 192.168.0.1

BGP neighbor is 192.168.0.1, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 15 seconds
 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.1
  eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%
  Connections established 0; dropped 0
  Last reset 00:03:23, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

Example 3: Neighbor group inherited value is used when higher precedence is absent

If the neighbor uses only a neighbor group (not a session group or address family group), and no value is set directly on the neighbor, the neighbor group value is used, either directly or through its own inheritance.

```
Router(config)# router bgp 150
Router(config-bgp)# session-group AS_2
Router(config-bgp-sngrp)# advertisement-interval 20
Router(config-bgp-sngrp)# exit
Router(config-bgp)# neighbor-group AS_1
Router(config-bgp-nbrgrp)# advertisement-interval 15
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# neighbor 192.168.1.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# use neighbor-group AS_1
```

This sample output from the **show bgp neighbors** command shows that the advertisement interval used is 15 seconds:

```
 Router# show bgp neighbors 192.168.1.1

BGP neighbor is 192.168.2.2, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 15 seconds

 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.1
  eBGP neighbor with no outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  Inbound path policy configured
  Policy for incoming advertisements is POLICY_1
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%

  Connections established 0; dropped 0
  Last reset 00:01:14, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

Example 4: Default value is applied if not configured or inherited

If the `advertisement-interval` is not configured directly or inherited, the system default (30 seconds) applies.

Additional notes

- The same inheritance and precedence rules apply when groups inherit values from other groups.
- For any inheritable property, always refer to the effective configuration displayed by the **show bgp neighbors** command to verify which value is ultimately used.

Result

These rules ensure that BGP neighbor and group configurations in Cisco IOS XR software are inherited and applied in a predictable order, allowing flexible and scalable BGP policy management.

Viewing inherited configurations

After applying configuration templates, it's important to verify which settings are active and inherited, using dedicated show commands such as **show bgp neighbors configuration**, **show bgp af-group**, and others. These sections provide sample outputs and tips for interpreting results.

Viewing BGP neighbor inheritance

Use the **show bgp neighbors** commands to view inherited configuration details for BGP neighbors. This table summarizes primary keywords, and their functions:

Command options

You can specify one of these keywords to refine the command output:

Keyword	Description
configuration	Displays the effective configuration for a BGP neighbor, including any inherited settings from session groups, neighbor groups, or address family groups.
inheritance	Shows the session groups, neighbor groups, or address family groups from which a neighbor can inherit configuration properties.

Sample BGP neighbor configuration

```
Router(config)# router bgp 142
Router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
Router(config-bgp-afgrp)# next-hop-self
Router(config-bgp-afgrp)# route-policy POLICY_1 in
Router(config-bgp-afgrp)# exit
Router(config-bgp)# session-group GROUP_2
Router(config-bgp-sngrp)# advertisement-interval 15
Router(config-bgp-sngrp)# exit
Router(config-bgp)# neighbor-group GROUP_1
Router(config-bgp-nbrgrp)# use session-group GROUP_2
Router(config-bgp-nbrgrp)# ebgp-multihop 3
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
Router(config-bgp-nbrgrp-af)# weight 100
Router(config-bgp-nbrgrp-af)# send-community-ebgp
Router(config-bgp-nbrgrp-af)# exit
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# neighbor 192.168.0.1
Router(config-bgp-nbr)# remote-as 2
Router(config-bgp-nbr)# use neighbor-group GROUP_1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# use af-group GROUP_3
Router(config-bgp-nbr-af)# weight 200
```

Command usage details

These commands allow you to view BGP neighbor configuration and inheritance details:

- **show bgp neighbors configuration**: Displays the active configuration for the specified neighbor, including inherited settings.
- **show bgp neighbors inheritance**: Shows which groups or templates provide inherited settings for the specified neighbor.

Viewing BGP address family group inheritance

Use the **show bgp af-group** command to view details about BGP address family groups, including details about configuration inheritance, the sources of inherited settings, and related entities that share configuration.

Command options

You can specify one of these keywords to refine the command output:

Keywords	Description
configuration	Displays the effective configuration for the address family group, including parameters inherited from other groups.
inheritance	Shows the address family groups from which the current group inherits configuration settings.
users	Lists the neighbors, neighbor groups, and address family groups that inherit configuration from the specified address family group.

Sample BGP address family groups configuration

```
Router(config)# router bgp 140
Router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
Router(config-bgp-afgrp)# remove-private-as
Router(config-bgp-afgrp)# route-policy POLICY_1 in
Router(config-bgp-afgrp)# exit
Router(config-bgp)# af-group GROUP_1 address-family ipv4 unicast
Router(config-bgp-afgrp)# use af-group GROUP_2
Router(config-bgp-afgrp)# maximum-prefix 2500 75 warning-only
Router(config-bgp-afgrp)# default-originate
Router(config-bgp-afgrp)# exit
Router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
Router(config-bgp-afgrp)# use af-group GROUP_3
Router(config-bgp-afgrp)# send-community-ebgp
Router(config-bgp-afgrp)# send-extended-community-ebgp
Router(config-bgp-afgrp)# capability orf prefix both
```

Example command outputs

- **configuration keyword:**

```
Router# show bgp af-group GROUP_1 configuration

af-group GROUP_1 address-family ipv4 unicast
  capability orf prefix-list both           [a:GROUP_2]
  default-originate                         []
  maximum-prefix 2500 75 warning-only       []
  route-policy POLICY_1 in                  [a:GROUP_2 a:GROUP_3]
  remove-private-AS                         [a:GROUP_2 a:GROUP_3]
  send-community-ebgp                       [a:GROUP_2]
  send-extended-community-ebgp              [a:GROUP_2]
```

This example shows the source of each configuration item. The default-originate command was configured directly on this address family group, as indicated by the brackets [ ]. The remove-private-as command was inherited from address family group GROUP_2, which itself inherited the setting from address family group GROUP_3.

- **users keyword:**

```
Router# show bgp af-group GROUP_2 users

IPv4 Unicast: a:GROUP_1
```

- **inheritance keyword:**

```
Router# show bgp af-group GROUP_1 inheritance

IPv4 Unicast: a:GROUP_2 a:GROUP_3
```

This example shows that the address family group GROUP_1 inherits settings directly from GROUP_2, which in turn inherits from GROUP_3.

Viewing BGP session group inheritance

Use the show **bgp session-group** command with the **inheritance** keyword to view how BGP session groups inherit configuration.

Command options

You can specify one of these keywords to refine the command output:

Keywords	Description
configuration	Displays the effective configuration for a session group, including inherited settings.
inheritance	Shows the session groups from which a given session group inherits configuration.
users	Lists the session groups, neighbor groups, and neighbors that inherit configuration from a specified session group.

Sample BGP session group configuration

```
Router(config)# router bgp 113
Router(config-bgp)# session-group GROUP_1
Router(config-bgp-sngrp)# use session-group GROUP_2
Router(config-bgp-sngrp)# update-source Loopback 0
Router(config-bgp-sngrp)# exit
Router(config-bgp)# session-group GROUP_2
Router(config-bgp-sngrp)# use session-group GROUP_3
Router(config-bgp-sngrp)# ebgp-multihop 2
Router(config-bgp-sngrp)# exit
Router(config-bgp)# session-group GROUP_3
Router(config-bgp-sngrp)# dmz-link-bandwidth
```

Example command outputs

- **configuration keyword:**

```
Router# show bgp session-group GROUP_1 configuration

session-group GROUP_1
 ebgp-multihop 2          [s:GROUP_2]
 update-source Loopback0 []
 dmz-link-bandwidth       [s:GROUP_2 s:GROUP_3]
```

- **users keyword:**

```
Router# show bgp session-group GROUP_3 users

Session: s:GROUP_1 s:GROUP_2
```

This example shows that both the GROUP_1 and GROUP_2 session groups inherit session parameters from the GROUP_3 session group.

- **inheritance keyword:**

```
Router# show bgp session-group GROUP_1 inheritance

Session: s:GROUP_2 s:GROUP_3
```

This example shows that the session group GROUP_1 inherits session parameters from the GROUP_3 and GROUP_2 session groups.

Viewing BGP neighbor group inheritance

Use the **show bgp neighbor-group** command to display detailed information about BGP neighbor group configurations, including inheritance and user relationships.

Command options

You can specify one of these keywords to refine the command output:

Keywords	Description
configuration	Displays the effective configuration for the neighbor group, including settings inherited from other groups.
inheritance	Displays the sources of configuration inheritance (address family groups, session groups, and neighbor groups) for the specified neighbor group.
users	Lists the neighbors and neighbor groups that inherit configuration from the specified neighbor group.

Sample BGP neighbor group configuration

```
Router(config)# router bgp 140
Router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
Router(config-bgp-afgrp)# remove-private-as
Router(config-bgp-afgrp)# soft-reconfiguration inbound
Router(config-bgp-afgrp)# exit
Router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
Router(config-bgp-afgrp)# use af-group GROUP_3
Router(config-bgp-afgrp)# send-community-ebgp
Router(config-bgp-afgrp)# send-extended-community-ebgp
Router(config-bgp-afgrp)# capability orf prefix both
Router(config-bgp-afgrp)# exit
Router(config-bgp)# session-group GROUP_3
Router(config-bgp-sngrp)# timers 30 90
Router(config-bgp-sngrp)# exit
Router(config-bgp)# neighbor-group GROUP_1
Router(config-bgp-nbrgrp)# remote-as 1982
```

```
Router(config-bgp-nbrgrp)# use neighbor-group GROUP_2
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
Router(config-bgp-nbrgrp-af)# exit
Router(config-nbrgrp)# exit
Router(config-bgp)# neighbor-group GROUP_2
Router(config-bgp-nbrgrp)# use session-group GROUP_3
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
Routerconfig-bgp-nbrgrp-af)# use af-group GROUP_2
Router(config-bgp-nbrgrp-af)# weight 100
```

Example command outputs

- **configuration keyword:**

```
Router# show bgp neighbor-group GROUP_1 configuration

   neighbor-group GROUP_1
    remote-as 1982                   []
    timers 30 90                     [n:GROUP_2 s:GROUP_3]
    address-family ipv4 unicast      []
     capability orf prefix-list both [n:GROUP_2 a:GROUP_2]
     remove-private-AS               [n:GROUP_2 a:GROUP_2 a:GROUP_3]
     send-community-ebgp             [n:GROUP_2 a:GROUP_2]
     send-extended-community-ebgp    [n:GROUP_2 a:GROUP_2]
     soft-reconfiguration inbound    [n:GROUP_2 a:GROUP_2 a:GROUP_3]
     weight 100                      [n:GROUP_2]
```

This example shows that the remote autonomous system is configured directly on neighbor group GROUP_1, and the send community setting is inherited from neighbor group GROUP_2, which in turn inherits the setting from address family group GROUP_3

- **users keyword:**

```
Router# show bgp neighbor-group GROUP_2 users

Session:      n:GROUP_1
IPv4 Unicast: n:GROUP_1
```

This example shows that the neighbor group GROUP_1 inherits both session-level (address family-independent) and IPv4 unicast configuration parameters from the neighbor group GROUP_2.

- **inheritance keyword:**

```
Router# show bgp neighbor-group GROUP_1 inheritance

    Session:      n:GROUP-2 s:GROUP_3
    IPv4 Unicast: n:GROUP_2 a:GROUP_2 a:GROUP_3
```

This example shows that the neighbor group GROUP_1 inherits session-level (address family-independent) configuration parameters from neighbor group GROUP_2, which in turn inherits its session parameters from session group GROUP_3. Additionally, GROUP_1 inherits IPv4 unicast configuration parameters from GROUP_2. GROUP_2, in turn, inherits these IPv4 unicast parameters from the GROUP_2 address family group, which itself inherits them from the GROUP_3 address family group.

BGP address family configuration and behavior

BGP can support multiple address families, such as IPv4 and IPv6, each of which must be explicitly enabled and configured. Proper address family configuration ensures correct routing behavior for various protocol families.

No default address family in BGP

- You must explicitly configure each required address family at the BGP router level for it to be active.
- To activate a BGP session for a neighbor under a specific address family, you must configure that address family under both the router and the neighbor configuration.
- You can configure a neighbor without any address family, but cannot configure an address family under a neighbor unless that address family is present at the global BGP router level.

Key Point:

No address family is enabled by default in BGP. Every address family must be explicitly defined both at the global router level and under the neighbor if required.

Neighbor address family combinations

- **Default VRF**

  In the default VRF, you can configure both:

  - IPv4 unicast
  - IPv4 labeled-unicast

  Both address families are supported and can be enabled under the same neighbor.

- **Non-default VRF**

  In non-default VRFs, configuring both IPv4 unicast and IPv4 labeled-unicast address families under the same neighbor is not supported.

  The router allows the configuration but displays this error:

  ```
  Router: %ROUTING-BGP-4-INCOMPATIBLE_AFI : IPv4 Unicast and IPv4 Labeled-unicast Address
   families together are not supported under the same neighbor.
  ```

  If both address families are present in a BGP session for a neighbor in a non-default VRF, behavior becomes unpredictable, which can result in:

  - Incorrect advertisement of prefixes
  - Reachability issues

Tip To avoid reachability issues, explicitly configure a route policy to advertise prefixes only through either IPv4 unicast or IPv4 labeled-unicast address families, not both.

Default address family for show commands

Many BGP operational (show) commands allow specifying the Address Family Identifier (AFI) and Subsequent Address Family Identifier (SAFI).

For protocol definitions, see RFC 1700 and RFC 2858.

Setting default AFI and SAFI

In Cisco IOS XR, you can define default AFI and SAFI values for parser commands, eliminating the need to specify them repeatedly.

Parser commands to set defaults:

- **set default-afi { ipv4 | ipv6 | all }**
- **set default-safi { unicast | multicast | all }**

Default values:

- AFI: **ipv4**
- SAFI: **unicast**

  To change these defaults, you must use the parser commands.

Note If you specify an AFI or SAFI in a show command, those supplied values override any parser defaults.

Checking current AFI or SAFI defaults

Use the **show default-afi-safi-vrf** command to view the current default AFI and SAFI settings.

Use case: Redistribute iBGP routes into IGP

In some advanced scenarios, you may need to redistribute iBGP-learned routes into an IGP, such as OSPF or IS-IS. This process should be approached with caution due to potential routing loops.

By default, iBGP routes are not advertised into IGPs. This procedure is required if you need iBGP routes dynamically included in your IGP's routing table.

Before you begin

- Ensure you are in privileged EXEC mode.
- Verify that BGP and your chosen IGP (OSPF or IS-IS) are already configured and operating.
- Understand the impact and risks of redistributing iBGP routes, particularly regarding potential routing loops.

Follow this step to redistribute iBGP routes into an IGP:

Procedure

Enter BGP router configuration mode, and enable redistribution of iBGP-learned routes into the IGP.

Example:

```
Router(config)# router bgp 120
Router(config-bgp)# bgp redistribute-internal
```

BGP Confederations

In very large networks, managing a full iBGP mesh can be challenging. BGP confederations address this by partitioning an AS into sub-ASs, improving scalability and simplifying management.

Characteristics of BGP routing domain confederations

BGP routing domain confederations are used to simplify iBGP mesh requirements and improve scalability in large autonomous systems. Key characteristics include:

- A confederation divides a single autonomous system (AS) into multiple sub-autonomous systems, called member ASes.
- The confederation appears as a single autonomous system to external BGP peers.
- Each member AS maintains a full internal BGP (iBGP) mesh within itself. It establishes only limited external BGP (eBGP-like) connections to other member ASes within the same confederation.
- Peers in different member ASes use eBGP sessions but exchange routing information as if they were iBGP peers within the confederation.
- Routing attributes such as next hop, multi-exit discriminator (MED), and local preference are preserved across confederation boundaries.
- A single Interior Gateway Protocol (IGP) can be used for the entire confederation.

This approach reduces the number of required iBGP sessions and simplifies network management while maintaining compatibility with external BGP peers.

Configure a BGP routing domain confederation

Reduce internal iBGP mesh complexity by grouping multiple autonomous systems (AS) into a BGP confederation, enabling easier management and scalability.

A BGP routing domain confederation divides a single autonomous system (AS) into multiple sub-autonomous systems (sub-ASes). This approach simplifies iBGP connectivity by fully meshing only within each sub-AS, while maintaining a consistent external BGP appearance. All sub-ASes in a confederation use the same IGP.

Before you begin

- Gather the confederation AS number (the confederation identifier).
- Collect the list of sub-AS numbers to include as confederation peers.
- Access the router in global configuration mode.

Procedure

Specify the confederation identifier, and associate all AS numbers that will participate in the confederation.

Example:

```
Router# configure
            Router(config)# router bgp 120
            Router(config-bgp)# bgp confederation identifier 5
            Router(config-bgp)# bgp confederation peers 1091
            Router(config-bgp)# bgp confederation peers 1092
            Router(config-bgp)# bgp confederation peers 1093
            Router(config-bgp)# bgp confederation peers 1094
```

The router now treats the configured sub-ASes as part of the same BGP confederation. To external peers, the group of sub-ASes appears as a single AS.

Example

This is a sample configuration showing several peers within a BGP confederation. The confederation is made up of three internal autonomous systems with AS numbers 6001, 6002, and 6003. To BGP speakers outside the confederation, it appears as a single autonomous system with AS number 666, as specified by the bgp confederation identifier command.

```
router bgp 6001
 bgp confederation identifier 666
 bgp confederation peers
  6002
  6003
   exit
 address-family ipv4 unicast
  neighbor 172.16.232.55
  remote-as 6002
   exit
 address-family ipv4 unicast
  neighbor 172.16.232.56
  remote-as 6003
   exit
 address-family ipv4 unicast
  neighbor 172.19.69.1
  remote-as 777
```

On a BGP router in autonomous system 6001, the bgp confederation peers command designates peers from autonomous systems 6002 and 6003 as special eBGP peers within the confederation. As

a result, peers at 172.16.232.55 and 172.16.232.56 receive updates with local preference, next hop, and MED attributes unchanged. In contrast, the router at 172.19.69.1 is a standard eBGP peer outside the confederation, so the updates it receives from this peer are treated as regular eBGP updates from autonomous system 666.

BGP confederation peerings

A BGP confederation peering is a routing relationship within a BGP confederation that:

- enables routers in different sub-autonomous systems (sub-AS) of the same confederation to exchange routing information
- allows specific route advertisements using iBGP while circumventing standard iBGP full mesh requirements, and
- provides mechanisms to override the split horizon rule and control route learning between peer routers.

Autonomous system (AS):

An autonomous system is a collection of routers under a single administrative domain that use Interior Gateway Protocols (IGPs) for internal routing and Exterior Gateway Protocols (EGPs), such as BGP, for inter-domain communication.

Sub-autonomous system (sub-AS):

A sub-autonomous system is a distinct subset within a larger autonomous system, with its own administrative control and routing policies. Sub-ASs are used within BGP confederations to simplify iBGP mesh requirements.

Confederation:

A BGP confederation divides a single large AS into multiple sub-ASs, reducing the iBGP mesh. Externally, the confederation appears as a single AS, while internally it consists of multiple sub-ASs. Routers within different sub-ASs peer using eBGP, but exchange routing information similar to iBGP sessions, preserving BGP path attributes.

Autonomous System Number (ASN):

The ASN uniquely identifies each autonomous system or sub-autonomous system in BGP routing.

Split horizon:

Split horizon is a BGP routing rule that prevents a router from advertising a route back into the sub-AS (or confederation) from which it was learned, thereby preventing routing loops.

Table 3: Feature History Table

Feature Name	Release Name	Description
Peering Between BGP Routers Within the Same Confederation	Release 7.11.1	You can now enable BGP peering between routers in the sub-autonomous system (AS) within a confederation to advertise specific router updates using iBGP. This capability ensures that the mesh of routers between sub-ASes in a confederation maintains consistent routing tables, ensuring proper network reachability. Enabling this feature helps improve preventing performance reduction and traffic management challenges. The feature introduces these changes: **CLI:** • `allowconfedas-in` **YANG Data Models** • New XPaths for `Cisco-IOS-XR-ipv4-bgp-cfg.yang` • `Cisco-IOS-XR-um-router-bgp-cfg` (see GitHub, YANG Data Models Navigator)

Challenge of full iBGP mesh in large networks

In large-scale networks, maintaining a full iBGP mesh within an autonomous system becomes impractical as the number of routers increases.

Role of BGP confederations

BGP confederations allow a single AS to be partitioned into multiple sub-ASs, each of which maintains a full iBGP mesh internally. Routers in different sub-ASs establish eBGP-like sessions (called confederation peerings), but inside the confederation, the attributes of iBGP are preserved.

Impact of the split horizon rule

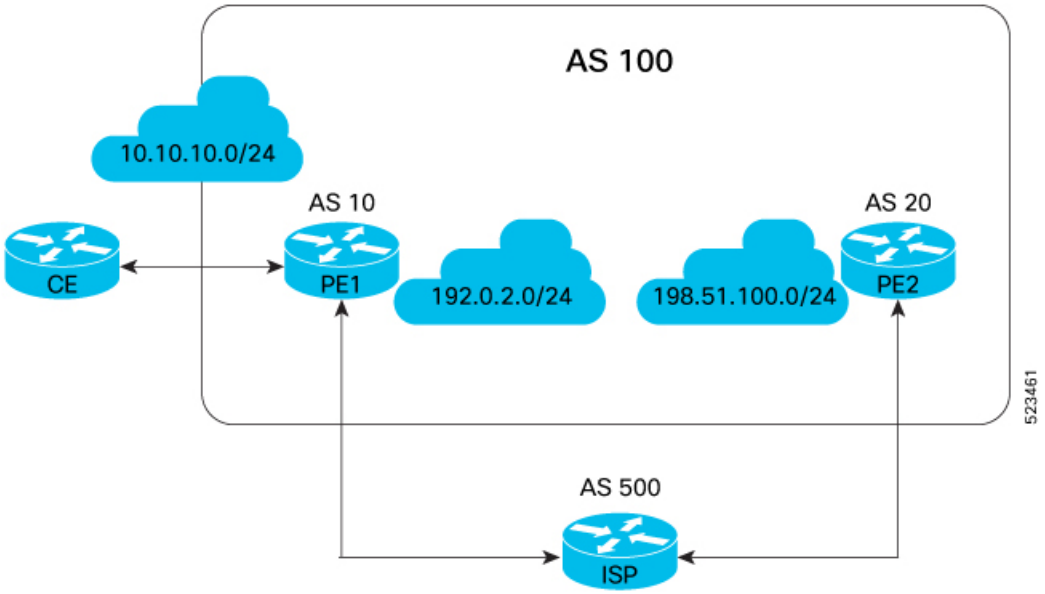
By default, the split horizon rule prevents routers in the same confederation from learning routes from one another if the route originated within the confederation.

Breaking split horizon

In scenarios requiring increased route flexibility or network customization, it may be necessary to break the split horizon rule. The **allowconfedas-in** command enables routers to bypass this restriction, allowing selected routes to be learned by peer routers within the confederation, and providing granular control over the number of times a route may be re-accepted within the confederation.

Example

Figure 1: BGP confederation peerings



In this sample topology, Router PE1 and Router PE2 are both part of a BGP confederation but belong to different sub-ASs (e.g., PE1 in sub-AS 100, PE2 in sub-AS 20). The CE router advertises the route 10.10.10.0/24 to PE1, which then advertises it to the ISP router (AS 500). The ISP router then passes the route to PE2. PE2 sees the confederation's AS numbers in the AS_PATH and, by default, drops the route due to the split horizon rule. To permit route learning in this scenario, configure the `allowconfedas-in` command on both PE1 and PE2. This allows PE2 to accept the 10.10.10.0/24 prefix from PE1, even though both routers are in the same confederation.

Limitations of configuring BGP confederation peerings

Configure allowconfedas-in only within specified limits

When you configure BGP confederation peering using the **allowconfedas-in** command, observe these limitations:

- Peer routers within a confederation can exchange information with each other only a limited number of times when the **allowconfedas-in** command is configured.
- The number of times information can be exchanged is limited to a range of 1 to 10.
- By default, the maximum number of exchanges is set to 3.

Configure BGP peering within a confederation

Enable peering between BGP routers that belong to the same BGP confederation, allowing them to exchange routing information.

Use this task when you want routers with different BGP autonomous system (AS) numbers, but within the same confederation, to peer and share routes.

Before you begin

- Ensure you are in privileged EXEC mode on each router.
- Confirm that you have the correct confederation AS numbers and that the routers are configured to use BGP.
- Identify the IP addresses of the routers you want to peer.

Follow these steps to configure BGP peering within a confederation:

Procedure

Step 1 Configure peer routers in the same confederation to learn from each other for a specified number of times.

Example:

```
Router# router bgp 65001
Router(config-bgp)# bgp confederation peers 65002
Router(config-bgp)# bgp confederation identifier 100
Router(config-bgp)# neighbor 198.51.100.3
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# allowconfedas-in 1
```

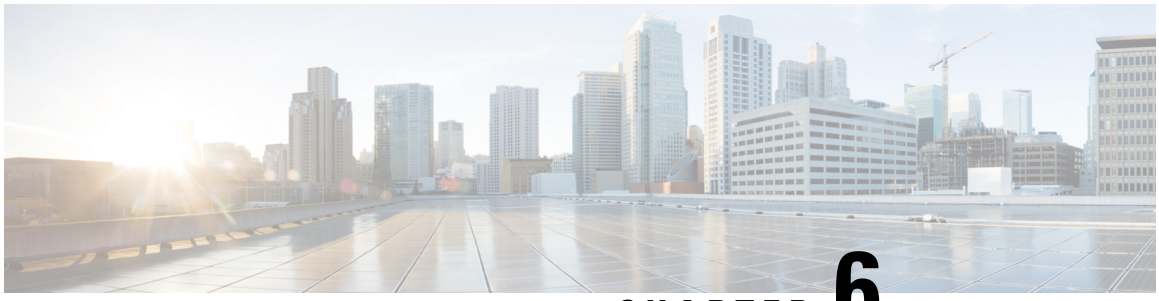
Step 2 Use the **show bgp neighbor** command to verify route learning among confederation peers.

Example:

```
show bgp neighbor 198.51.100.3 | in allow
Fri Mar  7 15:38:13.092 +0530
  Inbound soft reconfiguration allowed (override route-refresh)
  My confederation AS number is allowed 3 times in received updates.
```

This output shows that the peers within the same confederation have learned from each other's routes, and the learning among these peers has occurred three times.

The routers within the same BGP confederation are now configured as peers and can exchange routes, allowing repeated AS numbers in the received updates as specified.



CHAPTER 6

BGP Path Selection and Route Preference

BGP path selection is critical for efficient and policy-driven network routing. This chapter explains how BGP determines the best paths across complex networks and describes the configuration options available to influence route preference.

BGP best path algorithm

A BGP best path algorithm is a routing decision process that

- systematically evaluates multiple available BGP routes to the same destination based on a sequence of well-defined path attributes,
- selects the optimal route for installation in the routing table and advertisement to peers, and
- allows for administrative policy and configuration to influence path selection at various steps.

How BGP path selection works

BGP path selection occurs whenever multiple valid routes exist to the same destination. The router applies a series of steps, taking into account both protocol defaults and administrator-configured attributes to determine which path is installed in the routing table.

Summary

The key components involved in the process are:

- BGP router: A router that evaluates and compares multiple available routes using protocol-defined criteria.
- Configured attributes: Attributes that include weight, local preference, AS path, origin, MED, and community values, which influence path selection at each stage.

- BGP neighbors: External (eBGP) or internal (iBGP/ confederation), whose routes are candidates for selection.

When a router receives multiple BGP routes for the same destination, it selects the best path by first discarding invalid paths and preferring those with lower pre-bestpath cost communities if configured. It then compares attributes like weight, local preference, and whether the path is locally originated. Next, it evaluates AS path length, origin type, and MED among paths from the same AS. Advanced criteria favor eBGP-learned paths, lower IGP metrics to the next hop, and lower IP cost communities. If still tied, it uses tie-breakers such as lowest router ID, shortest cluster list, and lowest neighbor IP address. If all attributes tie, the current best path is retained until a change triggers re-selection.

Workflow

These stages describe the BGP path selection process.

1. The router receives multiple BGP route advertisements for the same destination from different peers.
2. The router performs preliminary checks:
   - removes invalid paths, such as those with unreachable next hops or exceeded MED values.
   - prefers the path with the lowest pre-bestpath cost community value.
3. The router compares attribute-based preferences and selects as follows:
   - Weight: chooses the path with the highest weight.
   - Local preference: prefers the path with the highest local preference value.
   - Locally originated path or aggregate: prefers any paths that are locally originated or created with the network or aggregate-address commands.
4. The router evaluates path characteristics:
   - AS path length: selects the path with the shortest AS path (unless the relevant configuration is set to ignore AS path).
   - Origin type: chooses the path with the lowest origin type (IGP over EGP over Incomplete).
   - Multi-Exit Discriminator (MED): selects the one with the lowest MED among paths from the same neighboring AS.
5. The router applies advanced criteria:
   - eBGP over iBGP: prefers a path learned via eBGP over those learned via iBGP or confederation peers.
   - IGP metric to next hop: chooses the path with the lowest IGP metric to the BGP next hop.
   - IP cost community: if present, selects the path with the lowest IP cost community value.
6. The router uses final tie-breakers when paths are still equal:
   - Router ID: selects the path received from the peer with the lowest router ID (using originator ID if available).
   - Cluster list length: prefers the path with the shortest cluster list.

- Neighbor IP address: As a last resort, chooses the path from the peer with the lowest neighbor IP address (locally generated paths are considered to have IP address 0).

7. If all these attributes result in a tie, the router maintains the current best path until a configuration change forces re-selection.

Example of path comparison order

Suppose a router receives three routes (A, B, C) to the same prefix. The router compares routes in sequence, stopping at the first attribute where one route is superior. The first attribute to break the tie determines the best path.

Order of comparisons and nontransitivity

The order in which BGP compares route attributes impacts routing decisions, especially because some attributes like MED are nontransitive and require special handling. These key points summarize this behavior:

- **MED nontransitivity:**
  - BGP compares the MED attribute only among routes received from the same neighboring autonomous system (AS). If route A is better than B, and B is better than C, route A is not necessarily better than C unless all three routes are from the same AS.
- **Grouping for MED comparison:**
  - Before evaluating the MED attribute, BGP groups routes by their neighboring AS. Within each group, BGP selects the route with the lowest MED value as the group's best route.
- **Iterative best path selection:**
  - Once a best-path is chosen from each group, BGP resumes the standard best-path selection process among these group winners to determine the overall best route.

Suppressing best path changes

BGP includes suppression logic to reduce unnecessary routing changes (churn) and enhance network stability.

Suppression logic:

Suppression logic operates under these conditions:

- If the new and current best paths are identical up to the decision point where tie-breakers (such as router ID) apply, BGP may retain the current best path to prevent route instability.
- Suppression generally applies only when both routes are learned from external peers and their routing attributes are otherwise equivalent.

Configuration note:

- Suppression can be disabled with the **bgp bestpath compare-routerid** command. When this command is used, any change in route attributes triggers a new best path selection.

Tuning BGP path selection

While BGP's default algorithm provides a robust method for selecting the best path, you must influence this decision to meet business or technical objectives. This section covers the major attributes and configuration tools available for tuning path selection, including the cost community, local preference, weight, MED, and the AIGP attribute.

BGP cost communities

A BGP cost community is a nontransitive extended community attribute that

- passes to internal BGP (iBGP) and confederation peers but not to external BGP (eBGP) peers,
- enables customization of local route preference by assigning cost values to specific routes, and
- influences the BGP best-path selection process at defined points in the algorithm.

The cost community feature allows network operators to prefer certain paths over others by assigning lower cost values. By default, the cost community affects the best-path algorithm after the IGP metric comparison. If multiple paths have equal preference, the cost community serves as a tie-breaker.

Points of insertion

The extended community format introduces generic points of insertion (POI) within the best-path selection process, allowing granular control over route preference. This mechanism gives administrators the flexibility to adjust the decision logic at key stages for more sophisticated routing policies.

How BGP cost communities affect best path selection

Cost communities can be configured to act as tiebreakers when several paths have equal preference by traditional BGP metrics. You can compare these attributes using a deterministic and hierarchical process.

Summary

The key components involved in the process are:

- **BGP router**: A router that analyzes multiple received paths and applies the cost community attribute during best-path computation.
- **Cost community attribute**: A path attribute that is used for fine-tuning the selection among equal-cost paths.
- **Point of Insertion (POI)**: The decision stage where cost community is evaluated before other standard metrics.

After IGP metric evaluation, the BGP router uses cost communities to compare paths. It sorts and compares attributes by point of insertion, community ID, and cost, preferring lower values. The router selects the path that ranks highest as the best path in the routing table.

Workflow

These stages describe how BGP cost communities affect best path selection:

1. Path reception: The BGP router receives multiple routes to the same destination.
2. POI evaluation: The router applies the point of insertion (POI), where cost community attributes take effect, after IGP metric comparisons.
3. Cost community comparison:

   When multiple paths have the cost community attribute, the BGP router compares them as follows:

   - Sort and prepare:
     - For each path, the router sorts cost community attributes by Point of Insertion (POI) and then by Community ID.
     - If a path does not include a cost community for the given POI and community ID, the router assigns it the default cost value (2147483647).
   - Pairwise comparison:
     - The router compares two paths at a time, examining the cost community attributes for the POI and community ID in order.
     - At each step:
       - If neither path has a relevant cost community, the router declares a tie for this comparison.
       - If only one path has a cost community, the router selects that path as the best path.
       - If both paths have cost communities with different community IDs, the path with the lower Community ID is chosen.
       - If both have the same Community ID but different cost values, the path with the lower cost value is selected.
       - If both have the same Community ID and cost value, the router continues to the next lowest Community ID and repeats the process.
     - The comparison ends when a best path is determined or all comparisons result in a tie.

Result

The BGP router installs the path with the most favorable cost community, so administrators can control routing decisions when standard BGP metrics do not produce a unique result.

Aggregate routes and multipaths

Aggregate routes and multipaths are extensions of the BGP cost community feature that

- allow the cost community attribute to be applied to aggregate and multipath routes
- ensure that unique cost community IDs from component routes are passed to the aggregate or multipath route, and
- apply only the highest cost value from all component routes for each unique ID to the aggregate or multipath route.

Inheritance of cost community attributes in aggregate and multipath routes

Aggregate or multipath routes inherit the BGP cost community attribute from the individual component routes. This means that any cost community attribute present in a component route is transferred to the aggregate or multipath route as part of the aggregation process.

Selection of cost values for each community ID

For each unique cost community ID found among the component routes, the aggregate or multipath route uses only the highest cost value for that ID. This ensures that, when multiple component routes share the same ID but have different cost values, the most significant (highest) value is selected and advertised.

Handling missing or mismatched cost community attributes

If a component route does not contain the cost community attribute, or if the component routes possess different cost community IDs, the default value of 2147483647 is advertised for those IDs.

Aggregate route with the same cost community ID

- Route 10.0.0.1 has POI=IGP, cost community ID=1, and cost number=100.
- Route 192.168.0.1 has POI=IGP, cost community ID=1, and cost number=200.

  After aggregation, the aggregate route advertises cost community ID=1 with a cost of 200, which is the highest value among the component routes.

Aggregate route with different cost community IDs

- Route 10.0.0.1 has POI=IGP, cost community ID=1, and cost number=100
- Route 172.16.0.1 has POI=IGP, cost community ID=2, and cost number=100
- Route 192.168.0.1 has POI=IGP, cost community ID=1, and cost number=200

  In this scenario, the aggregate route advertises the default cost value 2147483647 for both ID=1 and ID=2, since multiple IDs are present and cannot be consolidated under a single value.

If any component route does not include the cost community attribute, the aggregate or multipath route advertises the default cost value (2147483647) for that community ID.

Example: Adding routes to the Routing Information Base

The Routing Information Base (RIB) uses specific criteria for adding routes and selecting the best path. These criteria ensure the most optimal route is chosen based on BGP's cost community and best-path algorithms.

The RIB uses these criteria when adding routes and selecting the best path:

- If a nonsourced path becomes the best path after a best-path calculation, BGP adds the route to the RIB and passes cost communities along with other IGP extended communities.
- When a protocol adds a route with multiple paths to the RIB, the RIB checks both current best paths and incoming paths for the presence of cost extended communities.
- If cost extended communities are present:

- The RIB compares the sets of cost communities between paths.
- If there is a tie in cost community comparison, the RIB continues with the remaining steps of the best-path algorithm.
- If a cost community is missing from either current best paths or newly added paths, the RIB also continues with the remaining best-path evaluation steps.

Table 4: Summary:

Condition	RIB Action
Nonsourced path is best after calculation	Adds route to RIB with cost or IGP communities
Cost community found in compared paths	Compares cost communities, selects best
Tie in cost community comparison	Runs next steps of best-path algorithm
Missing cost community in any compared path	Runs next steps of best-path algorithm

Example: Influencing route preference in a multiexit IGP network

In large enterprise and service provider networks, it is common to have multiple exit points (border routers) connecting an internal network to one or more external networks or service providers. These are known as multiexit points. If interior gateway protocols (IGPs) such as OSPF or IS-IS provide equal-cost paths to these exit points, BGP treats all exit points equally.

Default behavior:

- If **multipath load sharing** is configured in BGP, both equal-cost paths are installed in the routing table and used for balancing traffic.
- If **multipath load balancing** is not configured, BGP selects the path that was learned first and installs only this path in the routing table. You might see suboptimal routing if, for example, the first-learned path uses a lower-speed link.

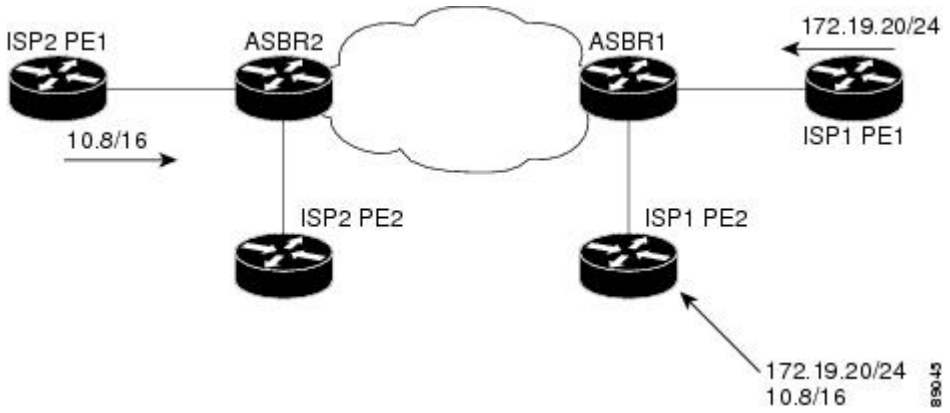
Using cost community to influence preference:

The BGP cost community attribute allows you to customize route preference within the AS, even when IGP metrics are equal. By assigning a lower cost community value to a preferred route, you can ensure that BGP selects that route over others.

Example:

This figure shows an IGP network with two autonomous system boundary routers (ASBRs) on the edge. Each ASBR has an equal cost path to network 10.8/16

Figure 2: Multiexit point IGP network



Suppose you want BGP to prefer the path learned from ASBR2 over ASBR1. You can apply a lower cost community value to the route received via ASBR2. For example:

```
Router(config)# route-policy ISP2_PE1
Router(config-rpl)# set extcommunity cost (1:1)
```

This configuration assigns a cost community number of 1 to the 10.8.0.0 route learned from ASBR2. By default, BGP sets the cost community value of the path learned from ASBR1 to 2147483647. Because 1 is lower than 2147483647, the path via ASBR2 is selected as the best path by BGP.

Configure a BGP cost community

Configure BGP with the cost community feature to enable tiebreaking among equal paths during the best-path selection process.

Before you begin

- Ensure you are in privileged EXEC mode on the router.
- Gather the desired route policy name and cost community information.

Follow these steps to configure a BGP cost community:

Procedure

Step 1 Enter route policy configuration mode, and set the BGP extended community attribute for cost in the route-policy.

Example:

```
Router# configure
Router(config)# route-policy costA
Router(config)# set extcommunity cost cost_A
```

Step 2 Configure BGP and apply the route policy to a BGP neighbor.

- Use the **default-information originate** command to originate default-information.
- Use the **aggregate-address** *[address | mask-length]* **route-policy** *route-policy-name* to configure an aggregate-address.

Example:

```
Router(config)# router bgp 120
Router(config-bgp)# default-information originate
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy costA in
```

Step 3 Use the **showbgp** *ip-address* command to verify the configuration.

Ensure that the cost community is displayed in this format:

Example:

```
Cost: POI : cost-community-ID : cost-number
```

Local preference and weight

Local preference and weight are key BGP attributes used to influence route selection within an AS, allowing for straightforward prioritization of certain paths.

Change the BGP default local preference value

Adjust the default local preference value for BGP paths to influence route selection.

Use this task when you need to make certain BGP paths more or less preferred in your network routing policy by changing the default preference value.

Before you begin

- Ensure you have administrative access to the router.

Follow this step to change the BGP default local preference value:

Procedure

Enter BGP router configuration mode for your autonomous system, and configure the default local preference value.

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# bgp default local-preference 200
```

This value overrides the default (100) for all BGP paths learned unless set otherwise.

Configure BGP weights

Assign a weight to routes received from a BGP neighbor to influence the best-path selection process.

BGP weights allow you to prefer specific neighbors for routing most of your traffic. By assigning a higher weight to routes learned from a particular neighbor, you control which path is chosen as the best for outgoing traffic.

Before you begin

- Determine the desired autonomous system (AS) number for your BGP router.
- Identify the IP address and remote AS number of the BGP neighbor whose routes you want to modify.

Follow this step to configure BGP weights:

Procedure

Configure the neighbor and assign a weight to all routes learned from the neighbor.

Example:

```
router# configure
Router(config)# router bgp 120
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# weight 41150
```

Multi-Exit Discriminator

The Multi Exit Discriminator (MED) is an attribute used in BGP to influence the path selection process when multiple exit points exist from one autonomous system (AS) to another. Understanding its comparison logic and configuration is important for controlling external routing behavior.

Conditions for BGP MED comparison

BGP compares the Multi-Exit Discriminator (MED) attribute under specific conditions controlled by default behavior and configuration options.

Table 5: MED comparison summary

Condition	MED compared?
Paths from same neighbor AS	Yes
bgp bestpath med always configured	Yes, for all paths
Paths from different neighbor AS (by default)	No
Paths with only confederation segments (default)	No
bgp bestpath med confed configured	Yes, among confederation peers

Default comparison rules

- BGP compares MED values only between paths received from the same neighboring AS by default.

  If two paths are from different neighboring ASes, MED is not considered unless specific configuration options are set.

- When the AS path starts with an AS_SEQUENCE, the neighbor AS is the first AS number in that sequence. MED is compared only with other paths from the same neighbor AS.
- If a path lacks an AS path or its AS path starts with an AS_SET, the path is considered internal, and MED is compared only with other internal paths.
- When a path contains only confederation segments or starts with confederation segments followed by an AS_SET, MED is not compared by default unless the **bgp bestpath med confed** command is configured.
- If the AS path starts with confederation segments followed by an AS_SEQUENCE, MED is compared with other paths whose neighbor AS matches the first AS in the AS_SEQUENCE.

Configuration overrides

- If the **bgp bestpath med always** command is set, MED is compared among all paths, regardless of the neighbor AS.
- If the **bgp bestpath med confed** command is set, MED is also compared for paths from confederation peers.

Handling missing MED values

- If a route does not include a MED attribute, BGP assigns it a default value of 0.
- When the **bgp bestpath med missing-as-worst** command is configured, a missing MED is treated as the highest possible value, making the path least preferred.

Key points

- Lower MED values are preferred. BGP chooses paths with the lowest MED when other path attributes are equal.
- MED comparison is only relevant if higher-priority path attributes (weight, local preference, AS path length, origin) are the same.

Configure MED for BGP routes

Set the MED value to advertise to BGP peers for routes that do not already have a MED attribute.

Before you begin

Ensure you are in global configuration mode on the router.

Procedure

Set the default metric (MED) for routes without an existing MED attribute:

Example:

```
Router# configure
Routing(config)# router bgp 120
Routing(config-bgp)# default metric 10
```

Accumulated IGP attributes for BGP

Accumulated IGP attributes for BGP are optional BGP path attributes that

- record and propagate the cumulative IGP metric along a BGP route
- enable BGP to make routing decisions based on IGP metrics, and
- support multi-AS environments by simulating IGP metric behavior across BGP domains.

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
Accumulated IGP Attribute for BGP	Release 7.3.2	This feature enables you to implement multiple contiguous BGP Autonomous Systems under a single administration. You can allow BGP to make its routing decisions based on the IGP metric just as an IGP would do.

The Accumulated IGP (AIGP) attribute is assigned an IANA type code and is included in BGP updates as a set of Type/Length/Value elements (TLVs) that specifically carry the accumulated IGP metric.

Purpose and benefits of AIGP

The AIGP attribute allows BGP to carry IGP cost equivalents between provider edge devices. It enables optimal path selection and OSPF-like behavior for redistributed routes within BGP.

Use case for AIGP in network design

AIGP is especially useful in networks where OSPF or LDP only carry metric information locally, but BGP redistributes prefixes or labels at area boundaries to remote areas. This design lets BGP make path decisions that include information about IGP costs from various network segments.

- When an OSPF or static route is redistributed into BGP, the AIGP attribute carries the total IGP cost from the originating router.
- Using a route-policy, the AIGP metric can be set during route redistribution to inform BGP of the correct path cost.

If the AIGP attribute is not set, BGP bases its routing decisions only on attributes like AS_PATH or LOCAL_PREF, without considering the underlying IGP costs. This can lead to suboptimal path selection in complex or multi-AS environments.

Configure accumulated IGP attributes for BGP

Enable BGP to use accumulated IGP (AIGP) metrics for route selection, simulating IGP metric-based decision-making and supporting seamless redistribution of IGP routes into BGP.

Use this task when you want BGP to compute routes based on IGP-derived metrics, especially in networks where prefixes are redistributed from IGP (such as OSPF) into BGP, and you need to preserve and propagate cost information throughout the BGP domain.

Before you begin

- Ensure BGP and the relevant IGP (such as OSPF) are configured and operational.
- Confirm that route-policies and address families relevant to redistribution are defined.

Follow these steps to configure accumulated IGP attributes for BGP:

Procedure

Step 1 Define a route-policy to set the AIGP metric using IGP cost.

Example:

```
Router(config)# route-policy aip_policy
Router(config-rpl)# set aigp-metric igp-cost
Router(config-rpl)# exit
```

Step 2 Configure redistribution of IGP routes (such as OSPF) into BGP using the defined route-policy.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# redistribute ospf route-policy aip_policy
```

Step 3 (Optional) Repeat for static routes or other sources as needed, applying a route-policy that sets or modifies the AIGP metric as appropriate.

Step 4 Use the **show bgp** *prefix* command to verify the configuration.

Example:

```
Router# show bgp 10.0.0.1
Thu Sep 30 21:21:15.279 EDT
BGP routing table entry for 10.0.0.1/32
Versions:
Process bRIB/RIB SendTblVer
Speaker 4694 4694
Last Modified: Sep 30 21:20:09.000 for 00:01:06
Paths: (2 available, best #1)
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
Local
192.168.0.1 (metric 2) from 192.168.0.1 (192.168.0.6)
Received Label 24000
Origin IGP, localpref 80, aigp metric 900, valid, internal, best, group-best, labeled-unicast
Received Path ID 1, Local Path ID 1, version 4694
Originator: 192.168.0.6, Cluster list: 192.168.0.1
Total AIGP metric 902 <-- AIGP attribute received.
```

Administrative distances

A administrative distance is a routing protocol metric that:

- quantifies the trustworthiness of a routing information source,
- determines which route is preferred when multiple protocols learn the same route, and
- assigns lower values to more trusted routes.

Role of administrative distance in route selection

When a router learns about a network route from more than one routing protocol (such as eBGP, OSPF, or EIGRP), it uses administrative distance to decide which route to install in the IP routing table. The protocol with the lowest administrative distance is favored.

Administrative distance and BGP path selection

Administrative distance does not influence the BGP path selection algorithm, but it does determine whether BGP-learned routes are installed in the IP routing table.

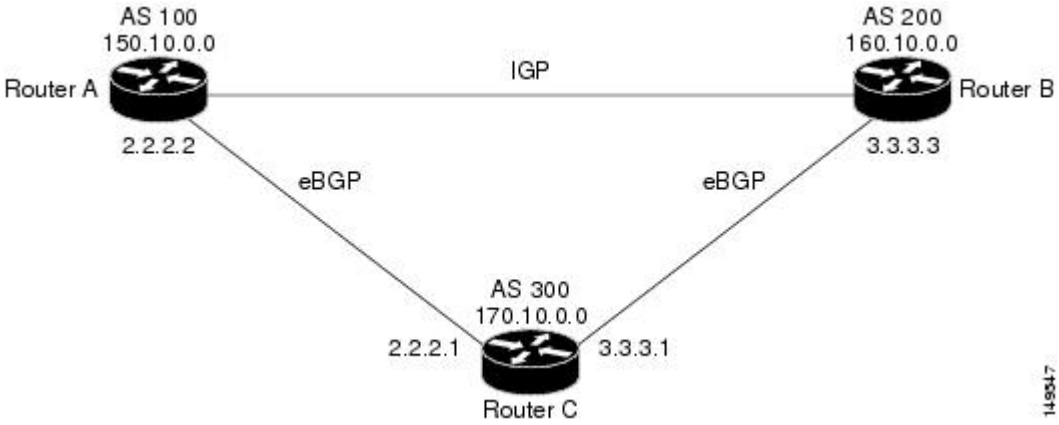
Default administrative distances for common protocols

In most cases, routes learned through eBGP (distance 20) are preferred over routes learned from IGPs (such as OSPF, distance 110; EIGRP, distance 90; IGRP, distance 100; and RIP, distance 120).

BGP default administrative distances

Distance	Default Value	Function
External	20	Applied to routes learned from eBGP.
Internal	200	Applied to routes learned from iBGP.
Local	200	Applied to routes originated by the router.

Figure 3: Back door example



Suppose Router A learns about network 160.10.0.0 from both eBGP (distance 20) and EIGRP (distance 90). The eBGP route is preferred because it has a lower administrative distance. If you want Router A to instead use the EIGRP path, you can configure a BGP back-door route:

```
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# network 160.10.0.0/16 backdoor
```

With this configuration, Router A treats the eBGP-learned route as a local route (distance 200), making the EIGRP-learned route (distance 90) the preferred path. If the EIGRP route fails, the eBGP route becomes active.

Configure BGP back-door routes

Configure a BGP back-door route so that the router prefers an IGP-learned path over an eBGP-learned path for specific network prefixes.

When a prefix is reachable via both IGP (internal) and eBGP (external), routers usually prefer the eBGP path due to its lower administrative distance. Use the BGP back-door feature to override this preference, so the router uses the IGP route and uses the eBGP route as a backup if the IGP path fails.

Before you begin

- Make sure BGP is enabled and running.
- Identify the network prefix you want to configure as a back-door route.

Follow this step to configure a BGP back-door route:

Procedure

Enter BGP configuration mode, and configure the network prefix with the backdoor option.

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# network 172.20.0.0/16 backdoor
```

Advanced path selection features

Modern BGP implementations provide advanced features for path diversity and resiliency. The BGP additional paths feature allows a BGP speaker to advertise more than just the best path, supporting use cases such as fast failover, route oscillation mitigation, and improved bandwidth utilization.

BGP best paths and multipaths

A best path and multipath prefix is a BGP routing outcome that

- identifies the optimal route (best path) for each destination prefix based on established selection rules
- enables the installation of multiple qualified paths (multipaths) in the forwarding table to enhance load balancing and redundancy, and
- determines which routes are advertised to BGP peers and how traffic is forwarded throughout the network.

BGP best path selection and advertisement

BGP routers receive multiple paths to the same destination prefix. By default, the BGP best path algorithm compares all valid paths and selects one as the best path to install in the local IP routing table and use for traffic forwarding. Only this best path is typically advertised to other BGP peers, which ensures consistent routing decisions across the network.

Multipath selection for load balancing and redundancy

However, to improve bandwidth utilization and provide redundancy, BGP implementations can select additional paths-called multipaths-if certain conditions are met. These multiple eligible paths are installed in the forwarding table, and incoming packets are load-balanced across the best path and the multipaths.

Community strings and the BGP add-path feature

Route reflectors in BGP can use community strings to signal local path decisions, differentiating between best paths and multipaths. The introduction of the BGP add-path feature allows a router to advertise more than the best path, signaling all equivalent or backup paths as per network policy and multipath rules.

Key considerations and limitations

- By default, BGP only advertises the best path to its peers. This behavior may not show you the entire routing state available on a BGP speaker.
- If Add-Path is not configured, installing multipath does not change what is advertised to peers.

A BGP router receives three paths to destination 10.1.1.0/24 from different peers. It selects one best path for routing and advertises only that path to other peers. If multipath is enabled and two paths qualify, both are used for forwarding traffic, providing load balancing and redundancy, though only the best path is advertised unless BGP add-path feature is configured.

Modify BGP best-path selection criteria

Change the default selection behavior of the BGP best-path algorithm to suit specific routing requirements.

BGP routers may learn multiple paths to a destination. By default, they use a specific algorithm to choose the best path. You can tune this behavior to influence routing and traffic flow between autonomous systems (ASes).

Before you begin

- Ensure you have access to global configuration mode on your router.
- Confirm that BGP is already enabled on the router.

Follow these steps to modify BGP best-path selection criteria:

Procedure

Step 1 Enter BGP configuration mode, and adjust MED comparison behavior.

- Use the **bgp bestpath med missing-as-worst** command to treat missing MED as least desirable.
- Use the **bgp bestpath med always** command to compare MED across all paths, regardless of AS.
- Use the **bgp bestpath med confed**command to compare MED values for confederation peers.

Example:

```
Router(config)# router bgp 126
Router(config-bgp)# bgp bestpath med missing-as-worst
```

Step 2 Customize path attribute evaluation.

- Use the **bgp bestpath as-path ignore** command to ignore AS path length for best-path selection.
- Use the **bgp bestpath compare-routerid** command to compare router IDs among otherwise equal paths.

Example:

```
Router(config)# router bgp 126
Router(config-bgp)# bgp bestpath as-path ignore
```

The BGP best-path algorithm selects routes based on your modified criteria.

BGP additional paths

A BGP additional path is a BGP feature that

- enables a BGP speaker to advertise multiple paths for the same network prefix
- provides path diversity to improve convergence and operational flexibility, and
- supports granular control of path selection and advertisement types per neighbor based on outbound policy configuration.

Table 7: Feature history table

Feature Name	Release Information	Feature Description
Additonal path control per neighbor	Release 7.3.15	This features allows flexibility and granular control of the advertisement of additional paths based on the neighbor outbound policy configuration. This is done by allowing configuration of combinations diff erent path selection procedures unlike singular path selection, and extending neighbor outpound policy to have finer control of the path types to be advertised. This feature enables operational efficiency to manage additional paths and reduce scale of the paths in a typical clustered network architecture. Without this feature, the path scale limitation of the memory is impacted, and control plane convergence issues develop because of the excessive number of paths.

The BGP additional paths feature modifies BGP protocol behavior so a BGP speaker can send more than one path for a given prefix.

Benefits for path diversity and convergence

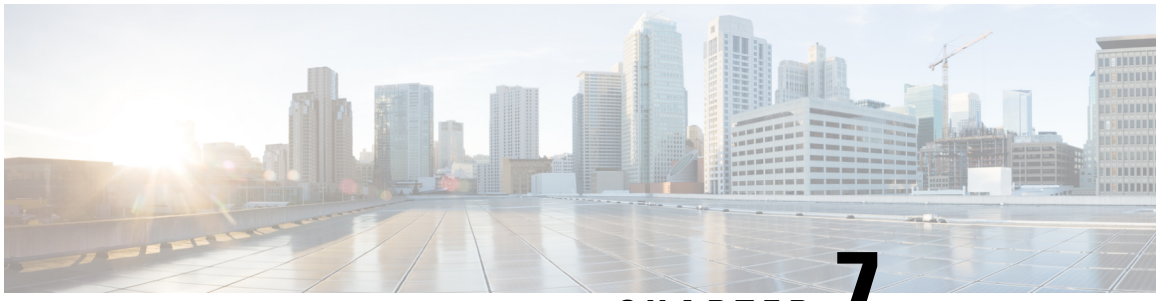
This modification provides network path diversity and enables edge routers to achieve prefix-independent convergence (PIC), especially in iBGP networks.

Enhanced policy control and scalability

By supporting multiple path types and expanded policy controls per neighbor, the feature helps prevent issues like path scale limitations and enables better control plane convergence.

BGP additional paths can advertise the following types of paths for a single prefix:

- Backup paths - enable fast convergence and rapid restoration of connectivity.
- Group-best paths - help resolve route oscillation scenarios.
- All paths - emulate an iBGP full mesh by advertising every possible path.



CHAPTER 7

BGP Routing Optimisation and Convergence Techniques

Border Gateway Protocol (BGP) is essential for inter-domain routing, but often presents challenges in scalability, path optimization, and convergence. This chapter introduces advanced BGP techniques designed to overcome these issues. We explore methods for enhancing route reflection, accelerating fault recovery, enabling intelligent path selection, and streamlining route management.

BGP route reflectors

A BGP route reflector is a type of internal BGP (iBGP) router that

- allows iBGP routers to share routes without needing every router to peer with every other router
- designates specific iBGP peers as clients to simplify network topology, and
- uses a cluster ID to coordinate with other route reflectors for redundancy and to prevent routing loops.

A **route reflector client** is an iBGP peer that receives routes from the route reflector and advertises its own learned routes back.

A **cluster** is a group consisting of a route reflector and its clients, and is identified by a unique cluster ID.

Route reflector mechanism:

In traditional iBGP networks, every iBGP router must peer with all other iBGP routers, which increases complexity as the network grows. BGP route reflectors address this scalability challenge by allowing designated routers (the route reflectors) to handle route sharing and peering. Only route reflectors maintain full iBGP sessions with each other, while other routers (clients) peer only with their assigned route reflector.

Redundancy with multiple route reflectors:

To increase redundancy and avoid a single point of failure, you can configure multiple route reflectors within the same cluster. Each reflector in the cluster uses the same 4-byte cluster ID. This ensures correct route learning and prevents routing loops.

Support for duplicate cluster IDs:

Special configuration options, such as the **cluster-id allow-equal** command, enables a router to accept routes with duplicate cluster IDs, but this option should be used carefully.

Configure a route reflector for BGP

Configure a router to function as a BGP route reflector and designate BGP neighbors as route-reflector clients within a specified cluster.

Before you begin

- Know the autonomous system (AS) number and desired cluster ID.
- Determine which BGP neighbors should be route-reflector clients.

Procedure

Configure a router as a BGP route reflector and configure the neighbor as its client.

Example:

```
Router(config)# router bgp 120
Router(config-bgp)# bgp cluster-id 192.168.0.1
Router(config-bgp)# neighbor 172.16.0.2
Router(config-bgp-nbr)# remote-as 65501
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-reflector-client
```

BGP multiple cluster IDs

A BGP multiple cluster ID is a route reflection feature that

- allows a route reflector to have both a global cluster ID and additional cluster IDs assigned to individual clients (neighbors)
- automatically adjusts the loop prevention mechanism based on CLUSTER_LIST to support multiple cluster IDs, and
- enables a network administrator to disable client-to-client route reflection based on cluster ID.

Before the introduction of multiple cluster IDs, a device could have only a single, global cluster ID. With this feature, configuring per-neighbor cluster IDs allows greater flexibility and control in BGP route reflection.

Remember The BGP multiple cluster IDs feature only works in the default VRF.

BGP optimal route reflectors

A BGP optimal route reflector (ORR) is a virtual route reflector function that

- calculates the best BGP path from the perspective of each route reflector (RR) client
- runs multiple shortest path first (SPF) calculations per RR client or cluster to ensure path optimality, and
- enables flexible placement of virtual route reflectors (vRR) in service provider networks without compromising path selection accuracy.

Traditional BGP route reflector limitations

In traditional BGP deployments, a route reflector acts as a focal point within an autonomous system and advertises routes to RR clients based on the RR's own path selection. When the RR is not optimally placed in the network topology, it can result in suboptimal routing decisions for RR clients, causing inefficient traffic flows.

Optimised client-specific routing with BGP ORR

BGP ORR addresses these limitations by running multiple SPF calculations from the perspective of each RR client or RR cluster. The system stores each client's SPF results in a dedicated database, using these results to influence BGP best path selection. This process ensures every advertised route is optimal for the client's specific network position, regardless of the vRR's location.

Benefits of BGP ORR

- Calculates and advertises the best BGP path for each RR client's viewpoint.
- Enables vRR placement anywhere in the service provider network without sacrificing routing efficiency.
- Allows network operators to scale RR memory and CPU resources according to operational requirements.

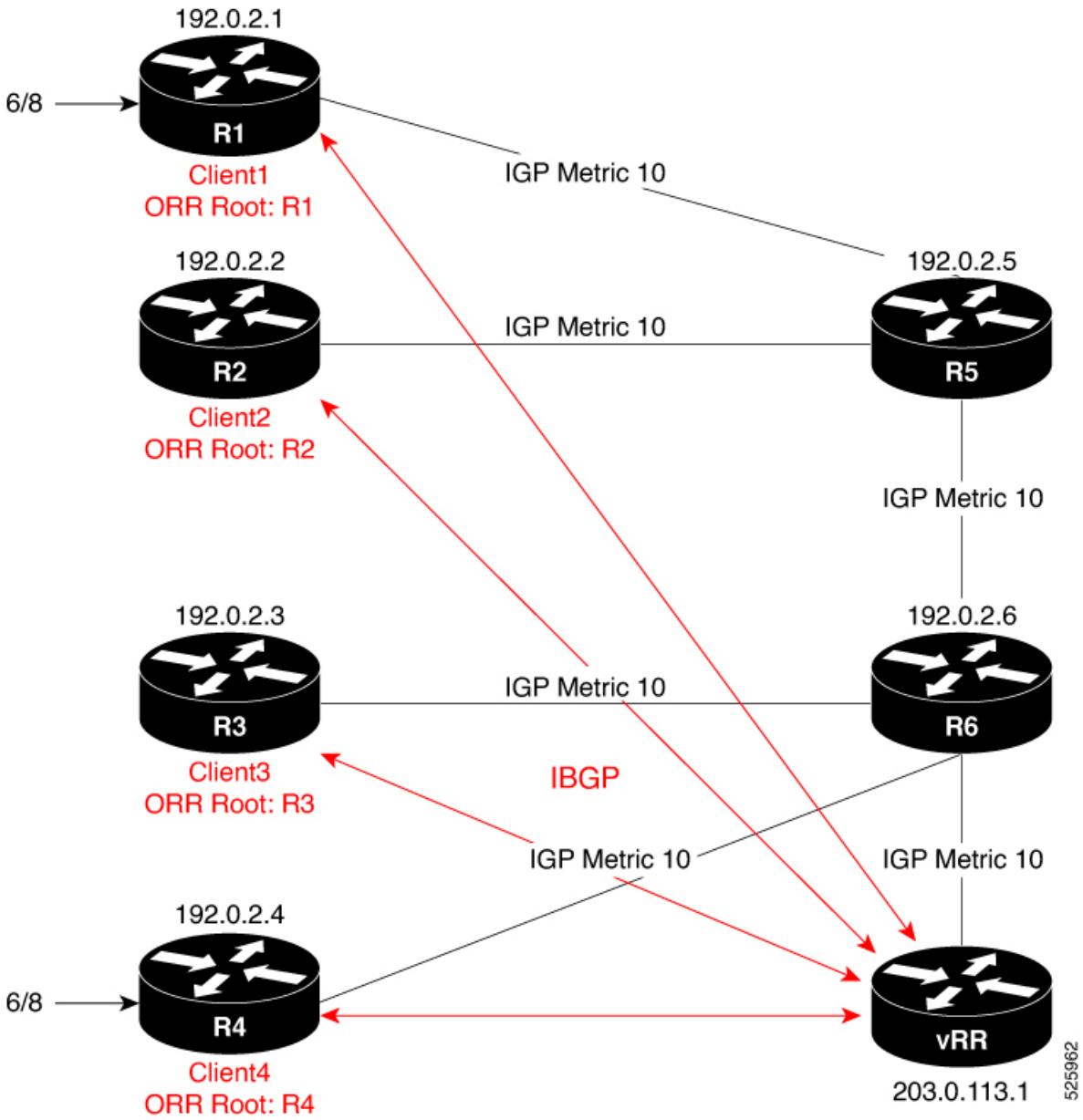
A service provider using network function virtualization (NFV) can deploy Cisco IOS XRv 9000 as a virtual route reflector in a central data center. BGP ORR optimizes routing for distributed RR clients, ensuring each receives the best available path despite the vRR's remote placement.

Effect of BGP ORR on route reflector client path selection

The path a route reflector client selects in a BGP topology depends on whether BGP ORR is configured.

Example: BGP ORR topology

To illustrate the impact of BGP ORR on client path selection, consider the topology shown in Figure 1.



In this sample BGP topology, routers R1, R2, R3, R4, R5, and R6 are route reflector clients connected to a virtual route reflector (vRR). R1 and R4 advertise the prefix 6/8 to the vRR. Without BGP ORR, the vRR reflects the prefix based on its own path selection, which may not be optimal for all clients. With BGP ORR enabled, the vRR selects the best exit for each client based on the client's location in the topology.

This table compares client path selection with and without BGP ORR:

Scenario	Client (Example: R2)	Exit point selected	Selection basis
Without BGP ORR	R2	R4	vRR selects best path from its own perspective

Scenario	Client (Example: R2)	Exit point selected	Selection basis
With BGP ORR configured	R2	R1	vRR calculates best exit from the client's perspective (ORR Root: R2)

Key facts

- **Without BGP ORR:**

  Route reflection uses the virtual route reflector's (vRR) view of the network topology, which may not yield the optimal exit point for every client. In this case, the vRR considers R4 as the best path for all clients, including R2, even if R2 is topologically closer to R1.

- **With BGP ORR:**

  The vRR evaluates topological distance from each client's perspective (the ORR root), ensuring each client receives the route that represents its optimal exit point. For example, R2 receives the route from R1 if it is the closest.

Configure BGP ORR for a route reflector client

Enable optimal route reflection (ORR) on a virtual route reflector (vRR) for a specific client, apply the correct policies, ensure underlying MPLS TE support, and verify correct function.

Before you begin

- Ensure you have administrative access to the vRR and root router.
- Confirm that BGP and MPLS Traffic Engineering features are enabled and licensed on all relevant devices.
- Gather these details:
  - BGP AS number
  - Client IP address
  - ORR root policy name
  - Required interface/loopback details

Procedure

Step 1 Enter BGP configuration mode and define the ORR statement using the client's IP address and the appropriate ORR policy.

Example:

```
Router# configure
Router(config)# router bgp 6500
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# optimal-route-reflection g1 192.0.2.2
Router(config-bgp-af)# commit
```

Step 2 In BGP configuration mode, assign the ORR policy to the target neighbor (the RR client) for the relevant address family.

Example:

```
Router# configure
Router(config)# router bgp 6500
Router(config-bgp)# neighbor 10.0.0.1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# optimal-route-reflection g1
Router(config-bgp-nbr-af)# commit
```

Step 3 On the root router, configure a minimal MPLS TE setup to advertise the router-ID that matches the configured root address on the vRR.

Example:

```
Router(config)# router isis 1
Router(config-isis)# is-type level-2-only
Router(config-isis)# net 47.0000.0000.0005.00
Router(config-isis)# distribute link-state
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# mpls traffic-eng level-2-only
Router(config-isis-af)# mpls traffic-eng router-id Loopback0
```

Step 4 Execute the **show bgp** command on the client (for example, R2) to verify whether the client received the best path.

Example:

```
R2# show bgp 10.0.0.0/8
Tue Apr  5 20:21:58.509 UTC
BGP routing table entry for 10.0.0.0/8
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 8           8
Last Modified: Apr  5 20:00:44.022 for 00:21:14
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    192.0.2.1 (metric 20) from 209.165.113.1 (192.0.2.1)
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best
      Received Path ID 0, Local Path ID 1, version 8
      Originator: 192.0.2.1, Cluster list: 203.0.113.1
```

Step 5 Execute the **show bgp** command on the vRR to verify that the client is in the correct update-group and that the best path is reflected as intended.

Example:

```
VRR# show bgp 10.0.0.0/8
Thu Apr 28 13:36:42.744 UTC
BGP routing table entry for 10.0.0.0/8
Versions:
Process bRIB/RIB SendTblVer
Speaker 13 13
Last Modified: Apr 28 13:36:26.909 for 00:00:15
Paths: (2 available, best #2)
Advertised to update-groups (with more than one peer):
0.2
Path #1: Received by speaker 0
ORR bestpath for update-groups (with more than one peer):
0.1
Local, (Received from a RR-client)
```

```
192.0.2.1 (metric 30) from 192.0.2.1 (192.0.2.1)
Origin incomplete, metric 0, localpref 100, valid, internal, add-path
Received Path ID 0, Local Path ID 2, version 13
Path #2: Received by speaker 0
Advertised to update-groups (with more than one peer):
0.2
ORR addpath for update-groups (with more than one peer):
0.1
Local, (Received from a RR-client)
192.0.2.4 (metric 20) from 192.0.2.4 (192.0.2.4)
Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best
Received Path ID 0, Local Path ID 1, version 13
```

Step 6 Execute the **show bgp update-group 0.1** command, and verify whether the client (R2) is in update-group 0.1.

Example:

```
VRR# show bgp update-group 0.1
Thu Apr 28 13:38:18.517 UTC

Update group for IPv4 Unicast, index 0.1:
Attributes:
Neighbor sessions are IPv4
Internal
Common admin
First neighbor AS: 65000
Send communities
Send GSHUT community if originated
Send extended communities
Route Reflector Client
ORR root (configured): g1; Index: 0
4-byte AS capable
Non-labeled address-family capable
Send AIGP
Send multicast attributes
Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 5, replicated: 5
All neighbors are assigned to sub-group(s)
Neighbors in sub-group: 0.2, Filter-Groups num:1
Neighbors in filter-group: 0.2(RT num: 0)
192.0.2.2
```

Step 7 Run the **show orrspf database** command to validate cost and root selection.

Example:

```
VRR# show orrspf database g1
Thu Apr 28 13:39:20.333 UTC

ORR policy: g1, IPv4, RIB tableid: 0xe0000011
Configured root: primary: 192.0.2.2, secondary: NULL, tertiary: NULL
Actual Root: 192.0.2.2, Root node: 2000.0100.1002.0000

Prefix Cost
203.0.113.1 30
192.0.2.1 20
192.0.2.2 0
192.0.2.3 30
192.0.2.4 30
192.0.2.5 10
```

```
192.0.2.6 20

Number of mapping entries: 8
```

BGP Accept Own

A BGP Accept Own mechanism is a BGP routing feature that

- allows a BGP speaker to accept VPN routes that it originally advertised but receives back from a route-reflector
- uses the special ACCEPT_OWN community attribute to bypass standard self-origination filters such as ORIGINATOR_ID and NEXTHOP checks, and
- enables centralized control of route imports across VRFs in MPLS VPN environments without requiring configuration changes on provider edge routers.

Default BGP route rejection behavior:

By default, according to BGP protocol (RFC 4271), a BGP speaker rejects routes it originated if they are returned by a route-reflector.

BGP Accept Own feature:

The BGP Accept Own feature changes this behavior: when the ACCEPT_OWN community is attached to a prefix (by a route-reflector via an outbound route-policy), it signals the receiving router to accept the reflected route even if it originally advertised that prefix. This is especially valuable in MPLS VPN extranets, where route import/export must be managed centrally.

Use case: Extranet auto-configuration in MPLS VPNs:

One primary use case is auto-configuration of extranets within MPLS VPNs. Conventionally, controlling route imports between VRFs for extranets requires updating import route-targets or policies on each PE router. With BGP Accept Own, route-reflectors can manage which prefixes are imported between VRFs without additional PE configuration. This approach makes operations more scalable and flexible.

Route-reflector handling and community propagation

Route-reflectors attach the ACCEPT_OWN community when advertising selected prefixes to the originating PE. The addition of this community should be limited to outbound policies targeting only the originator, to avoid unnecessary propagation. The InterAS route-reflector may also adjust the set of route targets (RTs) when sending routes with the ACCEPT_OWN community. This enables precise control over which VRFs receive the reflected routes.

Preference for Accept Own community in best path selection

Once the Accept Own community is attached to a route and propagated by a route reflector, remote PE routers apply these best path selection rules.

- The best path algorithm prefers a route that contains the Accept Own community over one that does not.

- This preference is evaluated immediately before the IGP metric comparison step in the best path selection process.
- If a remote PE receives an Accept Own path from one route reflector and a non-Accept Own path from another, and both paths are otherwise identical, it selects the Accept Own path.
- After a path is selected, the import process operates on the Accept Own path.

This behavior ensures that when multiple otherwise identical routes exist, routes with the Accept Own community are consistently preferred during best path selection on remote PEs.

How Accept Own routes are processed in BGP VPN configurations

Summary

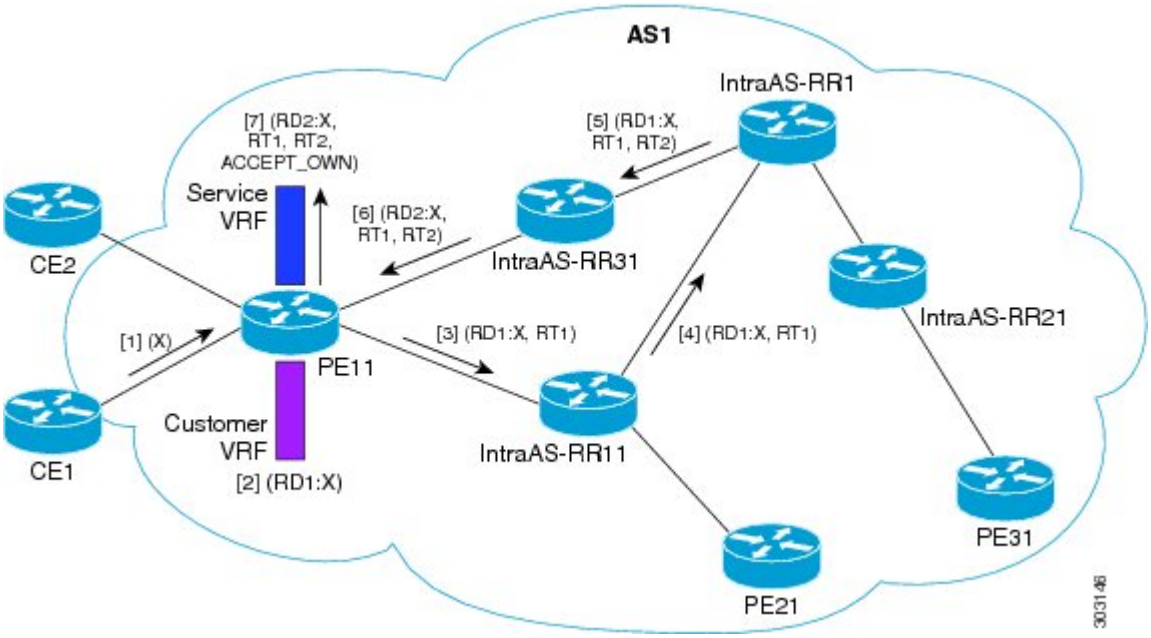
The key components involved in the process are:

- Customer edge (CE) router: A router that originates the desired prefix to the provider network.
- Provider edge (PE) router: A router that owns both the customer VRF and service VRF, participates in BGP VPN address families, and communicates with route reflectors.
- Intra-AS route reflector (IntraAS-RR): A route reflector (RR) that reflects routes within the same autonomous system.
- Inter-AS route reflector (InterAS-RR): An RR that adds the ACCEPT_OWN community and updates route targets before reflecting the route further.

A customer prefix is advertised from the CE to the PE, then reflected through multiple route reflectors. During this process, route targets and the ACCEPT_OWN community are added. The PE ultimately installs the prefix in the service VRF with updated attributes, enabling precise control over route import between VRFs.

Workflow

Figure 4: BGP Accept Own configuration example



These stages describe how Accept Own routes are processed in BGP VPN configurations:

1. Prefix origination: The CE router advertises prefix X to the PE.
2. Customer VRF installation: The PE installs prefix X in the Customer VRF as (RD1:X).
3. Initial advertisement to route reflector: The PE advertises prefix X to the IntraAS-RR as (RD1:X, RT1).
4. Inter-cluster reflection: The IntraAS-RR sends the route to the InterAS-RR as (RD1:X, RT1).
5. Community and route target update: The InterAS-RR attaches RT2 to prefix X on inbound and adds the ACCEPT_OWN community on outbound, then advertises the route to another IntraAS-RR.
6. Reflection back to PE: The second IntraAS-RR advertises the updated prefix to the PE.
7. Service VRF installation: The PE installs prefix X in the Service VRF with new route targets and the ACCEPT_OWN community (RD2:X, RT1, RT2, ACCEPT_OWN).

Result

The Accept Own process allows the same PE to import its own originated VPN prefixes into a different VRF. This supports extranet configurations with policy-driven control managed by the route reflector, without requiring modification of import policies or route targets on the PE.

Configure BGP Accept Own

Enable the BGP Accept Own feature so that the router accepts self-originated VPN routes containing the Accept_Own community attribute.

Before you begin

- Collect the required neighbor IP address, remote autonomous system number, and determine which address-family (VPNv4 or VPNv6) requires Accept Own.

Procedure

Step 1 Enter BGP configuration mode, specify the BGP neighbor, and enable Accept Own for the neighbor.

Example:

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# neighbor 192.0.2.3
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family vpnv6 unicast
Router(config-bgp-nbr-af)# accept-own
```

Step 2 (Optional) Use the **inheritance-disable** keyword to prevent inheritance of Accept Own from parent configuration.

Example:

```
Router(config-bgp-nbr-af)# accept-own inheritance-disable
```

This example shows how to configure BGP Accept Own on a PE router.

```
router bgp 100
neighbor 10.1.1.1
remote-as 100
update-source Loopback0 address-family vpnv4 unicast
route-policy pass-all in accept-own
route-policy drop_111.x.x.x out
!
address-family vpnv6 unicast route-policy pass-all in accept-own
route-policy drop_111.x.x.x out
!
!
```

This example shows an InterAS-RR configuration for BGP Accept Own.

```
router bgp 100
neighbor 192.0.2.45
remote-as 100
update-source Loopback0 address-family vpnv4 unicast
route-policy rt_stitch1 in route-reflector-client route-policy add_bgp_ao out
!
address-family vpnv6 unicast route-policy rt_stitch1 in route-reflector-client route-policy
 add_bgp_ao out
!
!
extcommunity-set rt cs_100:1 100:1
end-set
!
extcommunity-set rt cs_1001:1 1001:1
end-set
!
route-policy rt_stitch1
if extcommunity rt matches-any cs_100:1 then set extcommunity rt cs_1000:1 additive
endif
end-policy
!
```

```
route-policy add_bgp_ao
set community (accept-own) additive end-policy
!
```

Best-external paths

A best-external path is a BGP route selection mechanism that

- identifies the most optimal external path for a prefix when the current best path is internal (iBGP)
- enables advertisement of both the best and a backup path to peers to improve redundancy, and
- ensures internal routers are prepared with an external alternative if the primary iBGP path fails.

Selection procedure:

1. Determine the best path from all available paths for the prefix.
2. Remove the current best path.
3. Remove all remaining internal paths.
4. From the remaining paths, remove those with the same next hop as the current best path.
5. Use the BGP best path selection algorithm on the remaining paths to identify the best-external path.

Enhancing failover with the BGP best-external path feature

When a router's optimal path to a prefix comes from an iBGP peer, the router activates the best-external path feature to select and advertise the next best eBGP route to its internal peers. This function allows internal routers to promptly switch to a valid external route if the iBGP path becomes unavailable. As a result, the network avoids delays due to full path recalculation.

Configure best-external path advertisement

Enable the router to advertise the best-external BGP paths for a specific address family.

Before you begin

- Know your BGP autonomous system (AS) number.

Follow this step to configure best-external path advertisement:

Procedure

Enter global configuration mode, and enable best-external path advertisement.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn vpls-vpws
Router(config-bgp-af)# advertise best-external
```

BGP Prefix Independent Convergence

A BGP Prefix Independent Convergence (PIC) feature is a network routing enhancement that

- pre-computes and stores both primary and backup (best external) paths for each destination prefix in the Routing Information Base (RIB) and Forwarding Information Base (FIB)
- enables instantaneous switch to a backup path in the event of a primary path failure, minimizing convergence time and network downtime, and
- is especially beneficial for large-scale BGP deployments and networks with route reflectors.

Standard BGP convergence: prefix-dependent process

Standard BGP convergence is "prefix-dependent": each BGP router advertises only its current best path for a destination prefix. When the best path fails, routers send withdrawal messages, recalculate new best paths, and re-advertise to neighbors until all routers converge on a new path. This iterative process is slow, especially in networks with route reflectors.

Accelerated failover with Prefix Independent Convergence (PIC)

With PIC, BGP routers advertise their best external paths and pre-install backup paths in the FIB. When a failure occurs, the router can immediately switch to the backup path with a single operation, greatly accelerating convergence and reducing packet loss.

Attribute	Prefix-Dependent Convergence	Prefix Independent Convergence (PIC)
Awareness of backup paths	No	Yes
Convergence time	Slow, iterative	Fast, single-operation
FIB programming	Only best path	Best and backup paths
Scalability	Limited in large networks	Optimized for large deployments

Benefits of Prefix Independent Convergence

- Rapid recovery from link or node failures with minimal traffic disruption.
- Consistent failover times even as network scale grows.
- Improved reliability and service continuity for business-critical applications.

Selecting backup paths

Selecting backup paths ensures network resilience by identifying and programming an optimal alternative route in case the primary path fails.

Summary

The key components involved in the process are:

- Routing Information Base (RIB): A database that maintains all available routing information, including both primary and backup paths.
- Forwarding Information Base (FIB): A table that programs the selected backup path to forward data packets efficiently when needed.
- Best path algorithm: An algorithm that determines the best available path and the most suitable backup path from multiple candidates for each prefix.

The router uses the best path algorithm to select a primary path for a prefix. Next, the router removes the primary path and any paths that share its next hop. The algorithm runs again on the remaining paths to find the optimal backup path. The optimal backup is pre-programmed into the RIB and FIB for rapid failover if the primary path fails.

Workflow

These stages describe the process of selecting backup paths:

1. The router applies the best path algorithm to the available set of paths for a prefix to identify the primary (best) path.
2. The router excludes the best path from the set of available paths.
3. The router removes any paths that have the same next hop as the best path.
4. The router runs the best path algorithm again on the reduced set to select the optimal backup path.
5. The router programs the selected backup path into the RIB and FIB to ensure it is ready to take over if needed.

How prefix-independent convergence with route reflectors works

Prefix-independent convergence (PIC) with route reflectors optimizes BGP network failover by rapidly switching traffic to pre-programmed backup paths, minimizing convergence delays when a primary path fails.

Note To use the BGP PIC feature with route reflectors, each provider edge (PE) router must be configured with a unique route distinguisher (RD) within the same VRF. Without unique RDs, routes from different PEs appear to belong to the same network, preventing the route reflector from correctly identifying and calculating the best backup path.

Summary

The key components involved in the process are:

- Primary provider edge (PE) router: A PE router that advertises the primary route for traffic destined to remote PEs.
- Backup provider edge (PE) router: A PE router that maintains and advertises the backup (best external) path.
- Remote provider edge (PE) router: A PE router that selects between primary and backup paths based on announcements from core PEs.

- Route reflector: A router that distributes iBGP route information and requires unique route distinguishers (RDs) for proper path selection.
- Forwarding information base (FIB) or Routing information base (RIB): Stores and switches between primary and backup forwarding entries.

The process ensures fast BGP convergence by pre-installing both primary and backup paths in router forwarding tables (RIB and FIB). During normal operation, the primary path is used, while the backup path remains ready. If the primary path fails, the router instantly switches to the pre-programmed backup, enabling immediate traffic redirection with minimal delay and packet loss.

Workflow

These stages describe the process:

1. Pre-programming of paths: Both primary and backup (best external) routes for each prefix are programmed into the RIB and FIB of relevant routers.
2. Initial operation:
   - The primary PE advertises the **local-pref** attribute to designate itself as the preferred route; the backup PE advertises the backup path.
   - The remote PE receives both primary and backup paths but prefers the primary.
   - The route reflector uses unique route distinguishers to differentiate routes from different PEs within the VRF.
3. Primary path failure: In the event of primary path failure, primary PE signals the core to withdraw its route. The backup PE immediately advertises the backup path as the new best route.
4. Network convergence: The remote PE quickly recalculates and updates its primary path, switching from primary to backup.

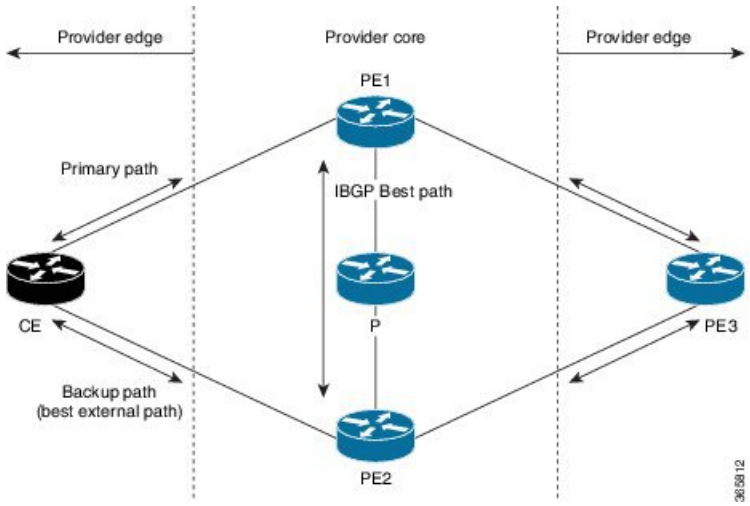
   The FIB instantly reassigns traffic to the backup path using the pre-installed forwarding entry for that prefix.
5. Traffic resumption: Network traffic resumes with minimal delay, as the backup route is already available in the FIB.

Configure BGP PIC in provider edge networks

Enable Prefix Independent Convergence (BGP PIC) to improve network resiliency by ensuring rapid failover between primary and backup paths in provider edge networks.

Consider this sample topology.

Figure 5: Prefix Independent Convergence in provider edge networks



For traffic traveling from the customer edge (CE) router to the provider edge (PE3) router, the BGP local preference (local-pref) attribute is used to determine the preferred path. The path CE → PE1 → PE3 is selected as the primary route, while CE → PE2 → PE3 is set as the backup route. Within the provider's core network, the path PE1 → P → PE2 is chosen as the best internal route between provider edge routers.

Before you begin

- Confirm all loopback and network interfaces are configured according to your topology.
- Ensure VRFs for the provider core network are set up.

Follow these steps to configure BGP PIC in provider edge networks.

Procedure

Step 1 On router PE1, configure BGP to install additional paths and set the label retention period.

Example:

```
Router(config)# router bgp 10
Router(config-bgp)# vrf foo
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# additional-path install
Router(config-bgp-vrf-af)# label-retention 10
```

Step 2 On router PE2, configure BGP to advertise the best external path, allocate labels, and enable additional path installation.

Example:

```
Router(config)# router bgp 10
Router(config-bgp)# vrf foo
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# advertise-best-external label-alloc-mode
Router(config-bgp-vrf-af)# additional-path install
```

Step 3 On router PE3, configure BGP to install additional backup paths.

Example:

```
Router(config)# router bgp 10
Router(config-bgp)# vrf foo
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# additional-path install
```

Step 4 On router PE3, verify that BGP PIC is operational. Verify the presence of backup paths in the Forwarding Information Base (FIB).

Example:

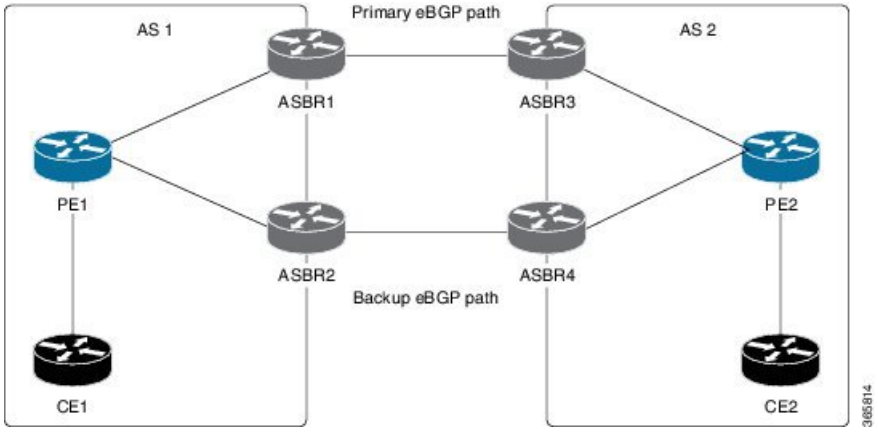
```
Router# show cef 10.1.1.1/32 detail
Fri Oct 10 10:24:33.079 UTC
10.1.1.1/32, version 1, internal 0x40000001 (0xa94c0574) [1], 0x0 (0x0), 0x0
(0x0)
Updated Oct 9 16:49:06.795
Prefix Len 32, traffic index 0, precedence routine (0)
gateway array (0xa8d9b130) reference count 4, flags 0x80200, source rib
(3),
[1 type 3 flags 0x901101 (0xa8ec6b90) ext 0x0 (0x0)]
LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
Level 1 - Load distribution: 0
[0] via 10.24.0.1, recursive
via 10.24.0.1, 3 dependencies, recursive
next hop 10.24.0.1 via 10.24.0.1/32
via 10.24.0.2, 3 dependencies, recursive, backup
next hop 10.24.0.2 via 10.24.0.2/32
Load distribution: 0 (refcount 1)
Hash OK Interface Address
0 Y MgmtEth0/RP0/CPU0/0 10.24.0.1
```

Configure BGP PIC Option B between autonomous systems

Configure BGP PIC for inter-AS Option B scenarios to ensure fast convergence and backup path installation between autonomous systems.

Consider this sample topology:

Figure 6: Prefix-Independent Convergence between autonomous systems



For traffic going from router PE1 to router PE2, ASBR1 acts as the primary router and ASBR2 as the backup router. The primary eBGP path is ASBR1 → ASBR3, while the backup path is ASBR2 → ASBR4.

For traffic traveling in the opposite direction, from router PE2 to router PE1, ASBR3 serves as the primary router and ASBR4 as the backup router. In this case, the primary eBGP path is ASBR3 → ASBR1, and the backup path is ASBR4 → ASBR2.

Before you begin

- Ensure that you have configured the loopback and network interfaces as per the illustrated topology.

Follow these steps to configure BGP PIC Option B between autonomous systems.

Procedure

Step 1 On router ASBR1, configure BGP additional-path and label retention.

Example:

```
Router(config)# router bgp 10
Router(config-bgp)# address-family vpnv4 unicast
Router(config-bgp-af)# additional-path install
Router(config-bgp-af)# label-retention 10
```

Step 2 On router ASBR2, configure advertisement and installation of backup paths.

Example:

```
Router(config)# router bgp 10
Router(config-bgp)# address-family vpnv4 unicast
Router(config-bgp-af)# advertise-best-external label-alloc-mode
Router(config-bgp-af)# additional-path install
```

Step 3 Similarly, repeat the above configuration steps on router ASBR3 for traffic from PE2 to PE1, and on Router ASBR4 to advertise and install backup paths for that traffic direction.

Step 4 Use the **show cef** command on router PE2 (for traffic from PE1 to PE2) or on router PE1 (for traffic from PE2 to PE1) to verify BGP PIC operation.

Example:

```
Router# show cef 10.1.1.1/32 detail

Fri Oct 10 10:24:33.079 UTC
10.1.1.1/32, version 1, internal 0x40000001 (0xa94c0574) [1], 0x0 (0x0), 0x0
(0x0)
Updated Oct 9 16:49:06.795
Prefix Len 32, traffic index 0, precedence routine (0)
gateway array (0xa8d9b130) reference count 4, flags 0x80200, source rib
(3),
[1 type 3 flags 0x901101 (0xa8ec6b90) ext 0x0 (0x0)]
LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
Level 1 - Load distribution: 0
[0] via 10.24.0.1, recursive
via 10.24.0.1, 3 dependencies, recursive
next hop 10.24.0.1 via 10.24.0.1/32
via 10.24.0.2, 3 dependencies, recursive, backup
next hop 10.24.0.2 via 10.24.0.2/32
Load distribution: 0 (refcount 1)
```

```
Hash OK Interface Address
0 Y MgmtEth0/RP0/CPU0/0 10.24.0.1
```

Step 5 Use the **show bgp vrf foo** command to verify the presence of the backup (best external) path for BGP.

Example:

```
Router# show bgp vrf foo 10.1.1.1/32

BGP routing table entry for 10.1.1.1/32
Versions:
Process bRIB/RIB SendTblVer
Speaker 6 6
Local Label: 3
Paths: (1 available, best #1)
Advertised to peers (in unique update groups):
10.10.10.1
Path #1: Received by speaker 0
10.1.1.1 from 10.1.1.1 (10.2.2.1)
Origin incomplete, metric 0, localpref 100, weight 32768, valid,
internal, best
10.2.2.2 from 10.2.2.2 (10.10.10.1)
Origin incomplete, metric 0, localpref 100, weight 32768, valid,
external, backup, best-external
```

Selective FIB download

A selective FIB download is a routing optimization feature that

- selectively installs specific destination routes in the Forwarding Information Base (FIB) rather than all available routes
- prevents traffic drops and black holes by ensuring traffic follows default routes when specific destination routes are unavailable, and
- improves network reliability in multi-data center environments by adapting installed routes to current topology and failures.

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
Selective FIB Download	Release 24.4.1	Introduced in this release on: Fixed Systems (8200, 8700); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q200]) You can now selectively download BGP prefixes to the Routing Information Base (RIB) and Forwarding Information Base (FIB). This feature prevents traffic drops by ensuring that traffic follows default routes when specific destination routes are unavailable.

Network challenges in multi-data center environments

In multi-data center networks, links or connections between intermediate distribution facilities (IDFs) and their termination points may occasionally fail. When these connections are unavailable, affected IDFs lose precise routing paths to target destinations, complicating traffic management and potentially disrupting service delivery.

Importance of aggregate routes in traffic management

To manage traffic efficiently, a main routing node distributes it across available paths using aggregate routes. An aggregate route is an aggregated network prefix representing multiple destinations within a specific network layer. Intermediate nodes rely on these aggregate routes to forward traffic toward the appropriate IDFs, based on network availability and routing policies.

Risks of traditional routing with missing specific routes

When traffic arrives at an IDF that lacks specific destination routes, the presence of aggregate routes on that IDF can misdirect traffic to a null interface. This situation creates a "black hole," where data packets are silently discarded, resulting in network connectivity problems and traffic loss.

Benefits of selective FIB download as a solution

Selective FIB download addresses these issues by preventing the installation of aggregate routes in the Routing Information Base (RIB) on local and connected intermediate nodes when direct routes to specific destinations are unavailable. Instead, the network relies on default routes to forward traffic toward alternative nodes that do possess the required specific routes. This approach ensures that traffic is intelligently rerouted through operational parts of the network, ultimately reaching its intended destination and avoiding black holes.

Address-family support

Selective FIB download supports IPv4 unicast and IPv6 unicast prefixes under the default VRF.

Configure selective FIB download

Configure selective FIB download to control which IP prefixes are installed or excluded from the router's Forwarding Information Base (FIB) by policy.

Before you begin

- Gather a list of IP prefixes that will be included in or excluded from the FIB.
- Determine which community values to use, such as comm_1 for install and comm_2 for exclude.

Follow these steps to configure selective FIB download.

Procedure

Step 1 Configure a prefix set with IP prefixes.

Example:

```
Router(config)# prefix-set route_install
Router(config-pfx)# 10.1.11.1/8
Router(config-pfx)# 2001:DB8:01::/32,
Router(config-pfx)# 10.3.3.3/8
Router(config-pfx)# 2001:DB8:FF::/32
Router(config-pfx)# end-set
Router(config-pfx)# exit
```

```
Router(config)# prefix-set route_not_install
Router(config-pfx)# 192.168.0.1/16
Router(config-pfx)# 2001:DB8:88::/32
Router(config-pfx)# 192.168.20.11/16
Router(config-pfx)# 2001:DB8:99::/32
Router(config-pfx)# end-set
Router(config-pfx)# exit
```

Step 2 Define and attach the community to the prefix set.

Example:

```
Router(config)# route-policy prefix_set_rpl
Router(config-rpl)# if destination in route_install then
Router(config-rpl-if)# set community comm_1
Router(config-rpl-if)# elseif destination in route_not_install then
Router(config-rpl-if)# set community comm_2
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
```

Step 3 Configure the route policy to specify which routes to install in the RIB and which routes to exclude.

Example:

```
Router(config)# route-policy rib_install_tb_rpl
Router(config-rpl)# if community matches-any comm_1 then
Router(config-rpl-if)# pass
Router(config-rpl-if)# elseif community matches-any comm_2 then
Router(config-rpl-if)# drop
Router(config-rpl-if)# else
Router(config-rpl-else)# pass
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
```

```
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy rib_install_tb_rpl
Router(config-bgp-af)# exit
Router(config-bgp)# address-family ipv6 unicast
Router(config-bgp-af)# table-policy rib_install_tb_rpl
Router(config-bgp-af)# exit
```

Step 4 Verify if the route is installed in RIB.

Example:

```
Router# show bgp ipv4 unicast 10.1.11.1/8
…
Paths: (1 available, no best path)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    10.2.2.2         10.1.2.3
  Local
    0.0.0.0 from 0.0.0.0 (10.1.1.1)
      Origin incomplete, metric 0, localpref 100, valid, local, permanent-path
      Received Path ID 0, Local Path ID 2, version 4
      Community: 11111:11111 22222:22222 33333:33333
      Origin-AS validity: not-found
```

Community: 11111:11111 22222:22222 33333:33333 in the sample output indicates the community **comm_1**.

Example:

```
Router# show route ipv4 10.1.11.1/8
Routing entry for 10.1.11.1/8
  Known via "bgp 65536", distance 200, metric 0, type locally generated
  Installed Jul 28 04:40:01.837 for 00:03:01
  Routing Descriptor Blocks
    directly connected, via Null0
      Route metric is 0
  No advertising protos.
```

Null0 in the sample output indicates that the route is installed in RIB.

These sample outputs illustrate scenarios where routes are not installed in the RIB.

Example:

```
Router# show bgp ipv4 unicast 192.168.0.1/16
Paths: (1 available, no best path)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192.168.2.2         192.168.1.2.3
  Local
    0.0.0.0 from 0.0.0.0 (192.168.1.1)
      Origin incomplete, metric 0, localpref 100, valid, local, permanent-path
      Received Path ID 0, Local Path ID 2, version 14
      Community: 44444:44444
      Origin-AS validity: not-found
```

Community: 44444:44444 in the sample output indicates the community **comm_2**.

Example:

```
Router# show route ipv4 192.168.0.1/16

% Network not in table
```

% Network not in table indicates that this route is not installed in RIB.

BGP-RIB feedback mechanisms

A BGP-RIB feedback mechanism is a route advertisement control feature that

- ensures that BGP announces routes to neighbors only after they are fully installed in the data plane
- relies on coordination between the Routing Information Base (RIB) and the Forwarding Information Base (FIB) to track route installation status, and
- prevents premature route advertisements that could otherwise cause packet loss during events such as router reloads or link failures.

Operation of the BGP-RIB feedback process

This mechanism works by having BGP wait for confirmation from the RIB that the installed routes are also present in the FIB. The RIB uses the BCDL feedback process to determine which versions of routes have been installed in the FIB and reports this information back to BGP.

Benefits of BGP-RIB feedback mechanisms

BGP only advertises routes that have been confirmed as fully installed, ensuring that network traffic is not directed to unprogrammed or unavailable paths. This enhances network reliability and prevents blackholing of traffic after topology changes.

Configure BGP to wait for feedback before sending updates

Ensure that BGP only advertises routes to neighbors after those routes are fully installed in the forwarding plane, preventing premature route announcements and possible packet loss.

Before you begin

- Ensure that changes to route advertisement timing are acceptable for your network's convergence policies.

Follow these steps to configure BGP to wait for feedback before sending updates to neighbors.

Procedure

Step 1 Use the **update wait-install** command to configure BGP to wait for RIB to confirm FIB installation before advertising routes.

Example:

```
Router# configure
Router(config)# router bgp 65500
Router(config-bgp)# address-family vpnv4 unicast
Router(config-bgp-af)# update wait-install
```

Step 2 Use the commands **show bgp**, **show bgp neighbors**, or **show bgp process performance-statistics** to verify and monitor the status.

BGP permanent networks

A BGP permanent network is a Border Gateway Protocol (BGP) feature that

- enables the creation and advertisement of selected prefixes (IPv4 or IPv6) using route-policies
- ensures these routes remain in the routing table until explicitly removed by an administrator, and
- allows the configuration and selective advertising of permanent paths to designated BGP peers.

Permanent path behavior

For each designated prefix, BGP establishes a permanent path. The permanent path is considered less preferred than dynamic BGP paths learned from peers and is only downloaded into the Routing Information Base (RIB) if it is evaluated as the best path. Permanent paths are selectively advertised and persist independently of changing network conditions.

Restrictions on BGP permanent networks

Review these restrictions before you configure BGP permanent networks:

- Use this feature only with IPv4 unicast and IPv6 unicast address families within the default Virtual Routing and Forwarding (VRF) context.
- Ensure that permanent paths are selectively advertised and persist only as described. Permanent paths do not override dynamic BGP paths unless you select them as the best path.

Configure a BGP permanent network

Configure a BGP permanent network to ensure designated prefixes are always advertised.

Use this task when you need certain prefixes to remain permanently available in BGP advertisements, regardless of current route availability.

Before you begin

- Ensure you have identified the IPv4 or IPv6 prefixes that should always be advertised.

Follow these steps to configure a BGP permanent network.

Procedure

Step 1 Define a prefix set for permanent network prefixes.

Example:

```
Router(config)# prefix-set PERMANENT-NETWORK-IPv4
Router(config-pfx)# 10.1.1.1/32,
Router(config-pfx)# 10.2.2.2/32,
Router(config-pfx)# 10.3.3.3/32
Router(config-pfx)# end-set
```

Step 2 Create a route policy to match the prefix set.

Example:

```
Router(config)# route-policy POLICY-PERMANENT-NETWORK-IPv4
Router(config-rpl)# if destination in PERMANENT-NETWORK-IPv4 then
Router(config-rpl)# pass
Router(config-rpl)# endif
```

Step 3 Enter BGP configuration mode, and configure the permanent network using the route-policy you created.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# permanent-network route-policy POLICY-PERMANENT-NETWORK-IPv4
```

Step 4 Use the **show bgp { ipv4 | ipv6 } unicast** command to confirm that the specified prefixes are advertised as permanent networks.

Advertise a permanent network

Configure BGP to ensure permanent network paths are always advertised to specified peers.

Use this task to maintain persistent route advertisement for critical network paths, even if the route is not present in the routing table.

Before you begin

- Identify the autonomous system numbers and the IP addresses of the BGP peers you want to configure.

Follow these steps to advertise a permanent network to BGP peers.

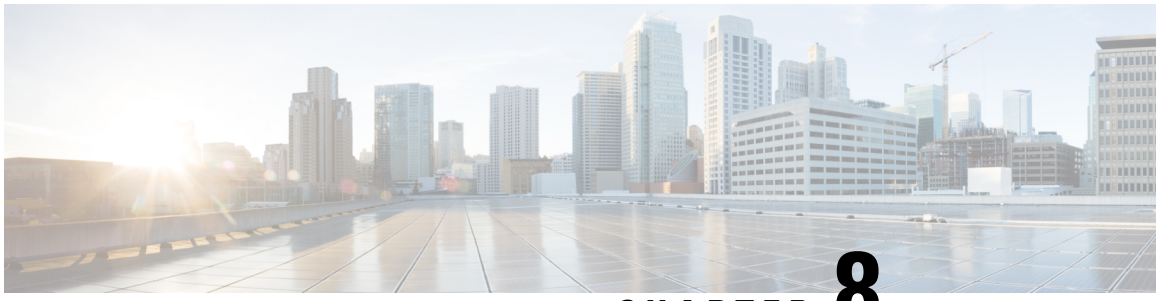
Procedure

Step 1 Enter BGP configuration mode, specify the neighbor IP address, and enable permanent network advertisement for that neighbor.

Example:

```
Router# configure
Router(config)# Router(config-bgp)# neighbor 10.255.255.254
Router(config-bgp-nbr)# remote-as 4713
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# advertise permanent-network
```

Step 2 Use the **show bgp {ipv4 | ipv6} unicast neighbor** *ip-address* command to to verify if permanent network advertisement is enabled.



CHAPTER 8

BGP Next-Hop Processing

This chapter explains how BGP handles next hop addresses for route processing, including notification mechanisms, configuration commands, route policies, and advanced use-cases such as MPLS LSP resolution.

BGP next hop notifications

A BGP next hop notification is a dynamic route monitoring mechanism that

- triggers updates to BGP route processing when a change in next hop reachability, connectivity, locality, or IGP metric is detected
- distinguishes between critical and noncritical event types for efficient network response, and
- supports policy-based filtering and batching to minimize route oscillation and ensure stable routing.

Table 9: Feature history table

Feature Name	Release Information	Feature Description
BGP Next Hop Tracking	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 88-LC1-36EH+A8:B12 • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M

Feature Name	Release Information	Feature Description
BGP Next Hop Tracking	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) The BGP next-hop tracking feature allows refined route resolution, avoiding aggregate routes and oscillation risks by filtering based on prefix length and source protocols, configurable through the nexthop trigger-delay and nexthop route-policy commands. * The BGP next hop tracking functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Event classification and handling

BGP classifies next hop events into two types:

- **Critical events:** These include changes in reachability (reachable or unreachable), connectivity (connected or disconnected), and locality (local or nonlocal). Critical events are processed and sent to BGP immediately to minimize convergence time.
- **Noncritical events:** These are changes in IGP metrics. Noncritical events are collected and sent in batches every three seconds. This batching helps minimize route churn and maintain network stability.

Events that trigger notifications

BGP receives next hop notifications when any of these events occur:

- A next hop becomes reachable or unreachable.
- A next hop becomes connected, unconnected, local, or nonlocal.
- The first hop IP address or interface changes.
- The recursed IGP metric for the next hop changes (e.g., due to network congestion or topology updates).

Configuration options for event handling

Several commands are available to fine-tune how BGP handles next hop notifications:

- **nexthop trigger-delay**: Adjusts the batching interval for noncritical events, per address family, allowing network operators to balance convergence speed with network stability.
- **nexthop route-policy**: Enables policy-based filtering of notifications to control which next hop changes are significant for BGP processing.

Example Scenarios:

- Critical Event: If an interface fails and a next hop becomes unreachable, BGP receives an immediate notification and recalculates the best path.
- Noncritical Event: When the IGP metric to a next hop increases due to network congestion, BGP receives a noncritical notification and may update its path selection after batching the event.

How BGP next hop notifications work

Summary

The key components involved in the process are:

- Routing Information Base (RIB): A database that continuously monitors the status and attributes of all next hops used by BGP.
- BGP process: A process that receives notifications from the RIB and updates routing information and neighbor relationships as needed.
- Network events: Events that trigger changes, such as interface status or IGP metric adjustments, that require BGP response.

The RIB continuously monitors BGP next hops and notifies BGP of any status changes. BGP then evaluates the event's impact, updates path selection, and recalculates next hop values to ensure correct routing.

Workflow

These stages describe the BGP next hop notifications process:

1. Monitoring: The RIB tracks all BGP next hops, continuously monitoring their reachability, connectivity, locality, and associated IGP metrics.
2. Detection: The RIB detects a change in next hop status such as an interface going down, a next hop becoming unconnected, or an IGP metric update.
3. Notification: Upon detecting a change, the RIB immediately notifies the BGP process about the specific event and affected next hop.
4. Evaluation: The BGP process verifies the new state of the next hop and determines whether the event is critical (immediate action needed) or noncritical (batching possible).
5. Update: BGP updates its path selection, recalculates outgoing next hop values for all affected routes, and checks or re-establishes neighbor connectivity as required.

Next hop determination (IPv4 and IPv6)

BGP uses a set of rules to determine the next hop address for prefixes exchanged between BGP peers. The determination logic differs depending on whether the prefixes are IPv4 or IPv6 and on the interface and neighbor configurations.

Next hops as IPv6 addresses of peering interfaces

When BGP is used for IPv6 routing, the next hop for IPv6 prefixes can be set to the IPv6 address of the peering interface. This behavior is defined as follows:

- **Assignment:**

  The IPv6 address of the peering interface is assigned as the next hop for IPv6 prefixes exchanged over an IPv4 BGP session.

- **Automatic application:**

  This assignment is automatically applied when no nexthop policy is configured but either an IPv6 neighbor or an IPv6 update-source interface is present.

- **Default behavior:**

  If neither an IPv6 neighbor interface nor an IPv6 update-source interface is configured, the next hop defaults to an IPv4-mapped IPv6 address.

IPv6 prefix transport over IPv4 BGP sessions

BGP supports the transport of IPv6 prefixes over IPv4 sessions. The determination of the next hop in these cases is controlled by the presence or absence of specific interface configurations and policies:

- With IPv6 neighbor or update-source configured:

  If an IPv6 neighbor interface or IPv6 update-source interface is configured, BGP uses the IPv6 address of that interface as the next hop.

- Without IPv6 neighbor or update-source:

  If neither interface is configured, BGP sets the next hop as an IPv4-mapped IPv6 address.

Table-policy for global IPv6 next hop

By default, BGP switches to a link-local IPv6 address for the next hop when ECMP links change, which may cause transient traffic loss. To maintain consistent load balancing and avoid these issues, configure a BGP table-policy to set a global IPv6 next hop.

Configure a BGP table-policy to set the global IPv6 next hop

Ensure BGP uses a global IPv6 address as the next hop and prevent traffic loss during ECMP path changes.

Before you begin

- Ensure you have access to the BGP route-policy configuration on your device.
- Ensure you have identified the destination prefixes that require consistent load balancing and global IPv6 next hop assignment.

Procedure

Define a new route-policy to set the global IPv6 address as the next hop.

Example:

```
Router(config)# route-policy RESILIENT-HASH-V6
Router(config-rpl)# if destination in (2001:DB8::/32 le 128) then
Router(config-rpl-if)# set load-balance ecmp-consistent
Router(config-rpl-if)# set next-hop ipv6-global
Router(config-rpl-if)# else
Router(config-rpl-else)# pass
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
```

Route resolution policies

Route resolution policies in BGP provide mechanisms to control how next hops are resolved and which routes are considered valid for advertisement or installation in the routing table. These policies offer fine-grained control over routing decisions and help prevent issues such as route oscillation or unwanted route propagation.

Policy criteria

BGP allows the creation and application of route policies that can specify:

- Prefix length requirements: For example, only allowing next hop routes with a prefix length greater than a configured value.
- Source protocol requirements: Requiring that next hop routes be learned from specific routing protocols (such as Intermediate System to Intermediate System (IS-IS), Open Shortest Path First (OSPF), or connected routes) to ensure routing stability.

Enabling route policy filtering

Enable route policy filtering in BGP using the **nexthop route-policy** command. You can apply this command globally, to specific address families, or at the VRF level, depending on your network design needs.

Scoped IPv4 table walks

A scoped IPv4 table walk is a route lookup mechanism that

- receives next-hop notifications to trigger address family processing
- uses gateway context information to determine which address families share the gateway context, and
- localizes processing to the IPv4 unicast address family table using a next-hop mask.

Identifying the address family from next-hop notifications

When a next-hop notification is received, the system first de-references the gateway context associated with that next hop. It then examines the gateway context to determine which address families are using it.

Gateway context sharing among IPv4 unicast address families

IPv4 unicast address families share the same gateway context, as they are registered with the IPv4 unicast table in the Routing Information Base (RIB). Each next-hop entry includes a mask to show whether it belongs to the IPv4 unicast address family.

Scoped table walk for efficient processing

Whenever a next-hop notification for IPv4 unicast is received from the RIB, the system processes only the global IPv4 unicast table. This scoped table walk ensures that updates or lookups are performed only in the relevant address family table, improving efficiency by avoiding unnecessary processing of unrelated address families.

Address family processing order

When a next-hop notification batch is received, the software reorders the address family processing in this order:

- IPv4 tunnel
- VPNv4 unicast
- IPv4 labeled unicast
- IPv4 unicast
- IPv4 multicast
- IPv6 unicast

This order determines how address family tables are walked based on their numeric value, ensuring efficient routing table updates in response to notifications.

Critical-event thread for BGP next hop processing

The dedicated critical-event thread in the spkr process handles next-hop, BFD, and fast-external-failover (FEF) notifications to help you achieve fast BGP convergence, even during other time-consuming events.

Configuration and commands

BGP provides a variety of configuration and operational commands to help users manage, monitor, and troubleshoot next hop processing. These commands allow you to control next hop handling in BGP updates, gather statistical information, and perform diagnostic analysis of next hop-related events.

Common BGP next hop commands

Show commands:

- **show bgp nexthops**:

  Displays statistical information about next hop notifications, processing time, and details about each next hop registered with the RIB. Use this command to monitor the status and performance of next hop processing.

Clear commands:

- **clear bgp nexthop performance-statistics**: Clears the cumulative statistics related to BGP next hop notification processing. Use this command when you want to reset the counters and start a new monitoring interval.

- **clear bgp nexthop registration**: Performs asynchronous registration of the next hop with the RIB. Use this command to reset and re-register next hops in case of inconsistencies or for troubleshooting purposes.

Debug commands:

- **debug bgp nexthop**: Provides diagnostic information on next hop processing.
  - The **out** keyword gives debug information about BGP registration of next hops with the RIB.
  - The **in** keyword displays debug information about next hop notifications received from the RIB.
  - If the **out** keyword is repeated, it displays debug information about next hop notifications sent to the RIB.

Disable next-hop processing on BGP updates

Ensure all BGP updates sent to a neighbor use the local router's address as the next hop, preventing automatic recalculation of next-hop values.

Disabling BGP next-hop processing is required when you want your router to appear as the next hop for all advertised routes to a peer. This is useful in designs where control over routing paths is necessary.

Before you begin

- Obtain your autonomous system (AS) number and the neighbor’s IP address.

Procedure

Configure the router to advertise itself as the next hop for all routes sent to the neighbor.

Example:

```
Router(config)# router bgp 120
Router(config-bgp)# neighbor 172.16.40.24
Router(config-bgp-nbr)# remote-as 206
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# next-hop-self
```

You can also disable next-hop processing for a neighbor group or an address family group, depending on your network design requirements.

VRF next hop route policies

A VRF next hop route policy is a routing control mechanism that

- enables you to configure route policies at the BGP next-hop attach point for individual VRF instances,
- limits notifications delivered to BGP for specific prefixes, and
- provides precise traffic engineering and security compliance for each VRF.

Table 10: Feature History Table

Feature Name	Release Name	Description
Virtual Routing Forwarding Next Hop Routing Policy	Release 7.11.1	You can now enable a route policy at the BGP next-hop attach point to limit notifications delivered to BGP for specific prefixes, which equips you with better control over routing decisions, and allows for precise traffic engineering and security compliance for each VRF instance, and helps establish redundant paths specific to each VRF. The feature introduces these changes: **CLI:** Modified Command: • The `nexthop route-policy` command is extended to VRF address-family configuration mode. **YANG Data Model** • New XPaths for `Cisco-IOS-XR-ipv4-bgp-cfg.yang` • `Cisco-IOS-XR-um-router-bgp-cfg` (see GitHub, YANG Data Models Navigator)

VRF next hop route policies give network administrators fine-grained control over route advertisement and notification within BGP processes. By assigning route policies to specific VRF address families, you can tailor routing behavior, enhance security, and ensure preferred routing paths for different tenants or services on your network.

Configure a VRF next hop policy

Enable and apply a next hop route policy to a VRF table. This allows you to control which routes are advertised to BGP peers based on prefix and protocol.

Use this task to ensure BGP only learns or advertises specific routes within a VRF.

Before you begin

Decide on the prefixes and protocols you want the route policy to match.

Procedure

Step 1 Define a route policy to match desired prefixes and protocols.

Example:

```
Router(config)# route-policy nh-route-policy
Router(config-rpl)# if destination in (10.1.1.0/24) and protocol in (connected, static) then
Router(config-rpl-if)# drop
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
Router(config-rpl)# exit
```

Step 2 Enter BGP configuration mode, and apply the next hop route policy to the VRF address family.

Example:

```
Router(config)# router bgp 500
Router(config-bgp)# vrf vrf10
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# nexthop route-policy nh-route-policy
```

Step 3 Use the **show bgp vrf** *vrf_name* **ipv4 unicast** command to verify if the policy is applied.

Example:

```
Router# show bgp vrf vrf1 ipv4 unicast
Fri Jul  7 15:51:16.309 +0530
BGP VRF vrf1, state: Active
BGP Route Distinguisher: 1:1
VRF ID: 0x6000000b
BGP router identifier 10.1.1.1, local AS number 65001
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe000000b   RD version: 1356
BGP table nexthop route policy: nh-route-policy --> This is the same route policy that was configured.
BGP main routing table version 1362
BGP NSR Initial initsync version 1355 (Reached)
BGP NSR/ISSU Sync-Group versions 1362/0

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
Route Distinguisher: 1:1 (default for vrf vrf1)
Route Distinguisher Version: 1356
*> 10.1.1.0/24      0.0.0.0       0       32768  ?
*> 192.0.2.0/24     10.1.1.1      0       32768  ?
```

BGP nexthop resolutions over MPLS LSPs with RSVP-TE tunnels

A BGP nexthop resolution over MPLS LSPs with RSVP-TE tunnels is a BGP feature that

- resolves BGP nexthops over RSVP-TE tunnels instead of native IP paths

- enforces controlled and predictable traffic steering by forwarding prefixes exclusively through MPLS LSPs, and
- prevents traffic drops caused by the lack of downstream BGP routing information in core networks.

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
BGP nexthop resolution over MPLS LSPs with RSVP-TE tunnels	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.

Feature Name	Release Information	Feature Description
BGP nexthop resolution over MPLS LSPs with RSVP-TE tunnels	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can now prevent BGP prefixes from defaulting to native IP paths, which could lead to traffic drops due to the lack of BGP routing information on downstream core routers, by enforcing BGP nexthop resolution over MPLS LSPs with RSVP-TE tunnels. The feature gives you precise control over traffic steering by defining how BGP resolves nexthops and enabling route policies that consistently forward prefixes over RSVP-TE tunnels. Previously, in core networks, downstream routers without BGP routes caused traffic to default to native IP paths instead of RSVP-TE LSPs, leading to potential drops. The feature introduces these changes: **CLI:** The **next-hop-type** is introduced as a filter type in RPL. **YANG Data Models:** `Cisco-IOS-XR-um-route-policy-cfg` (see GitHub, YANG Data Models Navigator)

BGP nexthop resolution over MPLS LSPs with RSVP-TE tunnels enables network operators to control BGP nexthop selection using route policies, ensuring prefixes are always forwarded over RSVP-TE engineered paths. This approach avoids traffic drops that can occur when downstream routers lack BGP routing information and traffic otherwise defaults to native IP paths. By integrating filtering based on route-type and next-hop-type, the feature optimizes path selection, prevents congestion, and supports predictable traffic engineering.

Importance in multihomed BGP environments

This feature is critical in multihomed BGP environments, where destination prefixes are reachable via multiple Label Edge Routers (LERs). It ensures that ingress LERs choose nexthops resolving over RSVP-TE LSPs, and not native IP paths, helping keep traffic within engineered tunnels.

Effect on traffic when RSVP-TE tunnels Are unavailable

When no RSVP-TE tunnel is available, BGP marks routes as inaccessible and drops packets, enforcing strict resolution over RSVP-TE tunnels. If a tunnel fails, BGP reroutes traffic through another available RSVP-TE tunnel to maintain continuous traffic flow.

Controlled traffic steering with RSVP-TE tunnels

Suppose a service provider core network has multiple downstream routers, some of which lack complete BGP information. With this feature enabled, BGP will resolve traffic only over RSVP-TE tunnels, which avoids unexpected drops even when native IP paths are present.

Benefits of nexthop resolution with RSVP-TE tunnels

Nexthop resolution over MPLS Label Switched Paths (LSPs) with RSVP-TE tunnels provides several key advantages in scalable network environments:

- Controls traffic steering by directing BGP prefixes over engineered traffic engineering (TE) paths, improving reliability and reducing packet drops.
- Enables flexible integration with multiple transport types and protocols, supporting seamless network expansion and future upgrades.
- Supports thousands of BGP nexthops and prefixes, maintaining high performance in large-scale deployments.

How BGP nexthop resolution over RSVP-TE tunnels works

Summary

The key components involved in the process are:

- BGP process: A process that exchanges routing updates for destination prefixes and determines next-hop addresses.
- Routing Information Base (RIB): A database that monitors next-hop resolution status and notifies BGP of changes, such as next-hop types or availability.
- RSVP-TE tunnels: A tunnel that establishes label-switched paths for traffic forwarding.

BGP uses RSVP-TE tunnels for nexthop resolution by applying route policies and selecting engineered paths. If an RSVP-TE tunnel becomes unavailable, affected routes are marked inaccessible and traffic is dropped until the tunnel is restored.

Workflow

These stages describe the BGP nexthop resolution over RSVP-TE tunnels process.

1. The BGP process receives routing updates for destination prefixes and attempts to resolve each prefix's next-hop address.
2. The RIB monitors the status of next-hop resolution and notifies BGP of any changes, including a change in next-hop type (such as resolution over an RSVP-TE tunnel).

3. BGP evaluates available paths by applying configured route policies and selects those that can be resolved over RSVP-TE tunnels.
4. The router forwards packets based on the best path as determined by BGP, steering traffic over the RSVP-TE tunnels specified.
5. If the RSVP-TE tunnel for a route's next-hop becomes unavailable, BGP marks the route as inaccessible and the router drops traffic for affected prefixes until resolution is restored.

Best practice for configuring BGP nexthop resolution over RSVP-TE tunnels

When configuring BGP nexthop resolution over RSVP-TE tunnels, follow these best practices:

- Ensure that route policies with extended filters are applied to the VPNv4, VPNv6, IPv4, and IPv6 address families.
- Apply the feature to all supported routes within the relevant address families to maintain consistent policy enforcement.

Configure BGP nexthop resolution with RSVP-TE route policy

Limit BGP nexthop resolution to RSVP-TE tunnels by applying a custom route policy.

Before you begin

Ensure RSVP-TE tunnels are established and operational within your network.

Procedure

Step 1 Define a route policy that permits only nexthops resolved using MPLS-TE.

Example:

```
Router(config)#route-policy ROUTE-RESOLUTION
Router(config-rpl)#if protocol is isis 100 and route-type is level-1 then
Router(config-rpl-if)#pass
Router(config-rpl-if)#elseif protocol is isis 100 and route-type is level-2 and next-hop-type is
mpls-te then
Router(config-rpl-elseif)#pass
Router(config-rpl-elseif)#else
Router(config-rpl-else)#drop
Router(config-rpl-else)#endif
Router(config-rpl)#end-policy
Router(config)#commit
```

Step 2 Apply the route policy to the BGP address-family configuration.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# nexthop route-policy ROUTE-RESOLUTION
Router(config-bgp-af)# commit
Router(config-bgp)# address-family ipv6 unicast
Router(config-bgp-af)# nexthop route-policy ROUTE-RESOLUTION
Router(config-bgp-af)# commit
```

Step 3 Use the **show bgp ipv4 unicast nexthops** command to verify if BGP resolves the nexthop over RSVP-TE tunnels.

Example:

```
Router# show bgp ipv4 unicast nexthops 209.165.200.225
Tue Mar 18 04:47:32.759 UTC
Nexthop: 129.134.99.7
  VRF: default
  Nexthop ID: 0x6000004, Version: 0
  Nexthop Flags: 0x00000000
  Nexthop Handle: 0x7f8d5fab66a8
  Tree Nexthop Handle: 0x7f8d5fab66a8

  RIB Related Information:
  Firsthop interface handle 0x7800004c
    Gateway TBL Id: 0xe0000000    Gateway Flags: 0x00000080
    Gateway Handle: 0x7f8d5d8b9d10
    Gateway: reachable, non-Connected route, prefix length 32
    Resolving Route: 129.134.99.7/32 (isis 1)
    Paths: 0
    RIB Nexthop Route Type: ISIS level-2
    RIB Nexthop Path Type: MPLS-TE
    RIB Nexthop ID: 0x0
    Nexthop sync slot: 15
    Status: [Reachable][Not Connected][Not Local]
    Metric: 30
    ORR afi bits: 0x0
    Inactive Tables: [IPv6 Unicast]
    Registration: Asynchronous, Completed: 00:27:39
    Events: Critical (3)/Non-critical (0)
    Last Received: 00:01:21 (Critical)
    Last gw update: (Crit-notif) 00:01:21(rib)
    Reference Count: 10

    Reachable Notifications:           1 (last at Mar 18 04:19:54.340)
    Unreachable Notifications:         0
    Metric Increase Notifications:     0
    Metric Decrease Notifications:     2
    Nexthop find:                      8
    Most Recent Events:
      Time                 EventType    Metric   Ifhandle     RouteType           PathType
      Mar 18 04:19:54.340  Reachable    30       0x7800004c   ISIS level-2        MPLS-TE
      Mar 18 04:44:47.974  Reachable    30       0x78000014   ISIS level-2        any
      Mar 18 04:46:12.411  Reachable    30       0x7800004c   ISIS level-2        MPLS-TE

  Prefix Related Information
    Active Tables: [IPv4 Unicast]
    Metrices: [0x1e]
    Reference Counts: [10]
    Encapsulations: []
  Interface Handle: 0x0
  Attr ref-count: 13
```

Review the output for `PathType MPLS-TE`, indicating successful resolution over RSVP-TE tunnels.

Step 4 Use the **show bgp ipv4 unicast** command to verify BGP routing information and nexthop reachability.

Example:

```
Router# show bgp ipv4 unicast 209.165.201.1/27
Thu Feb 13 12:14:35.626 UTC
BGP routing table entry for 209.165.201.1/27
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker           38992497     38992497
```

```
Last Modified: Feb 13 12:14:25.298 for 00:00:10
Paths: (1 available, no best path)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  64800, (received & used)
    10.1.9.7 (inaccessible) from 10.3.9.7 (10.4.9.7)
      Origin IGP, localpref 100, valid, internal
      Received Path ID 0, Local Path ID 0, version 0
      Community: 65530:50700
```

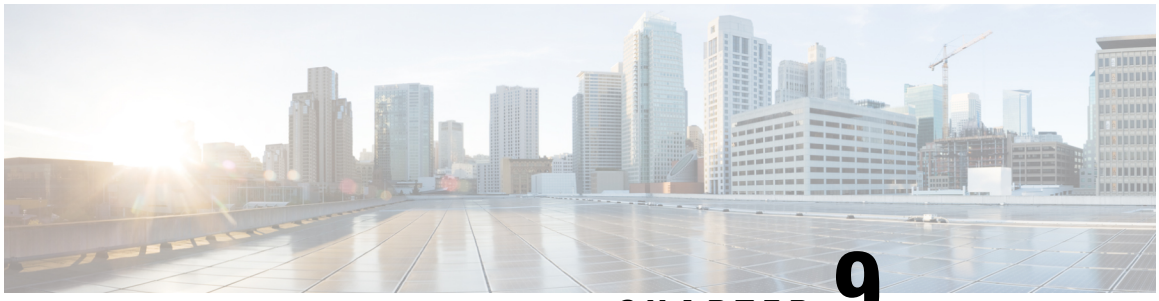
If no RSVP-TE tunnel exists, the nexthop appears as `inaccessible`.

Step 5 Use the **show rib ipv4 unicast next-hop** command to confirm if the RIB is forwarding the route over MPLS-TE tunnels.

Example:

```
Router#show rib ipv4 unicast next-hop 209.165.201.1/27

Tue Mar 18 04:47:33.806 UTC

Firsthop prefix: 209.165.201.1/27
  Flags: allow default, recurse
  Ext flags: 0x1 (all_path_mpls_te)
  Damped counter: 0
  Damp algo hits: 3
  Last event occurred Mar 18 04:46:12.409, 00:01:21 ago; version 4

  Registered clients:
    te_control/node0_RP0_CPU0 created Mar 18 04:19:23.479, 00:28:10 ago
       read last notification at Mar 18 04:46:12.411, 00:01:21 ago
       reference count is 1

  Destination paths:
    209.165.201.1 - S-AR1-DR1-1-1
    209.165.201.1 - S-AR1-DR1-2-1
    209.165.201.1 - S-AR1-DR1-3-1
    209.165.201.1 - S-AR1-DR1-4-1
    209.165.201.1 - S-AR1-DR1-5-1
    209.165.201.1 - S-AR1-DR1-6-1
    209.165.201.1 - S-AR1-DR1-7-1
    209.165.201.1 - SF-AR1-DR1-1-1

  Resolving route: 209.165.201.1/27 known via "isis 1"
  Metric computed: 30
```

If `Ext flags: 0x1 (all_path_mpls_te)` is present, the route is using only MPLS-TE tunnels.



CHAPTER 9

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 12: 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:

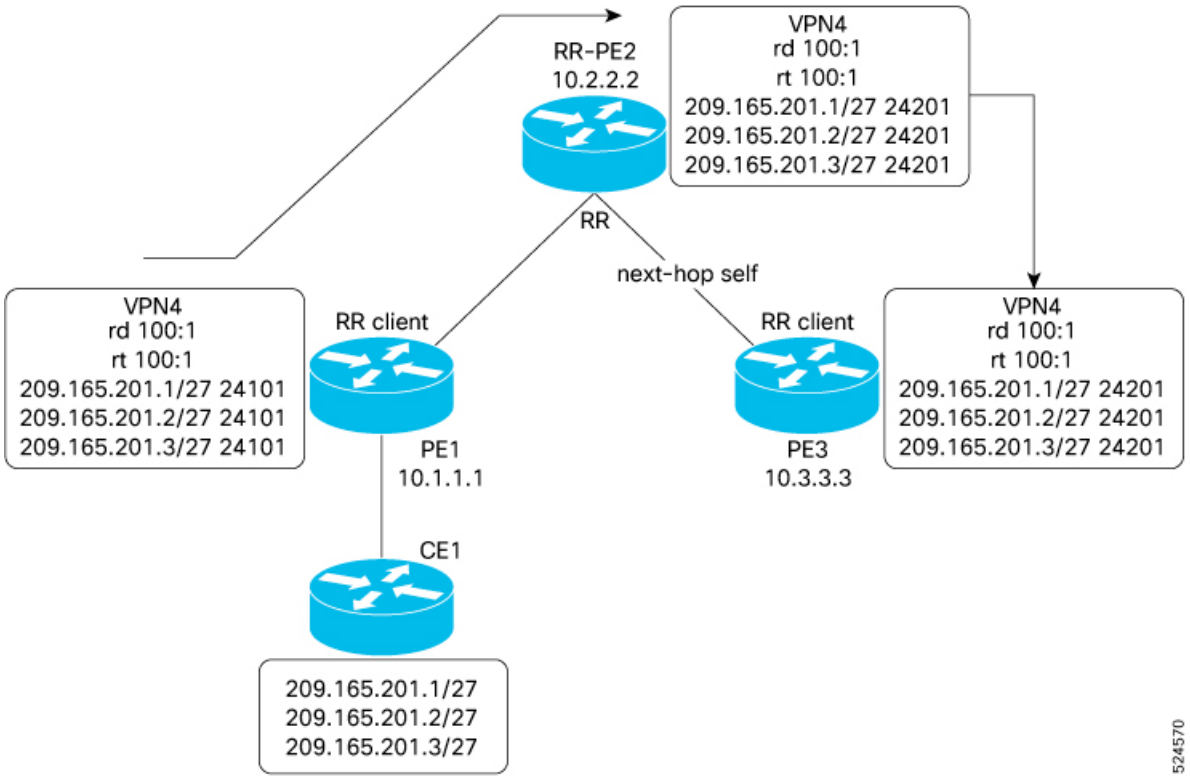
- Scenario 1: RR and PE routers are configured with the same RD
- Scenario 2: RR and PE routers are configured with different RDs

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 7: 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:

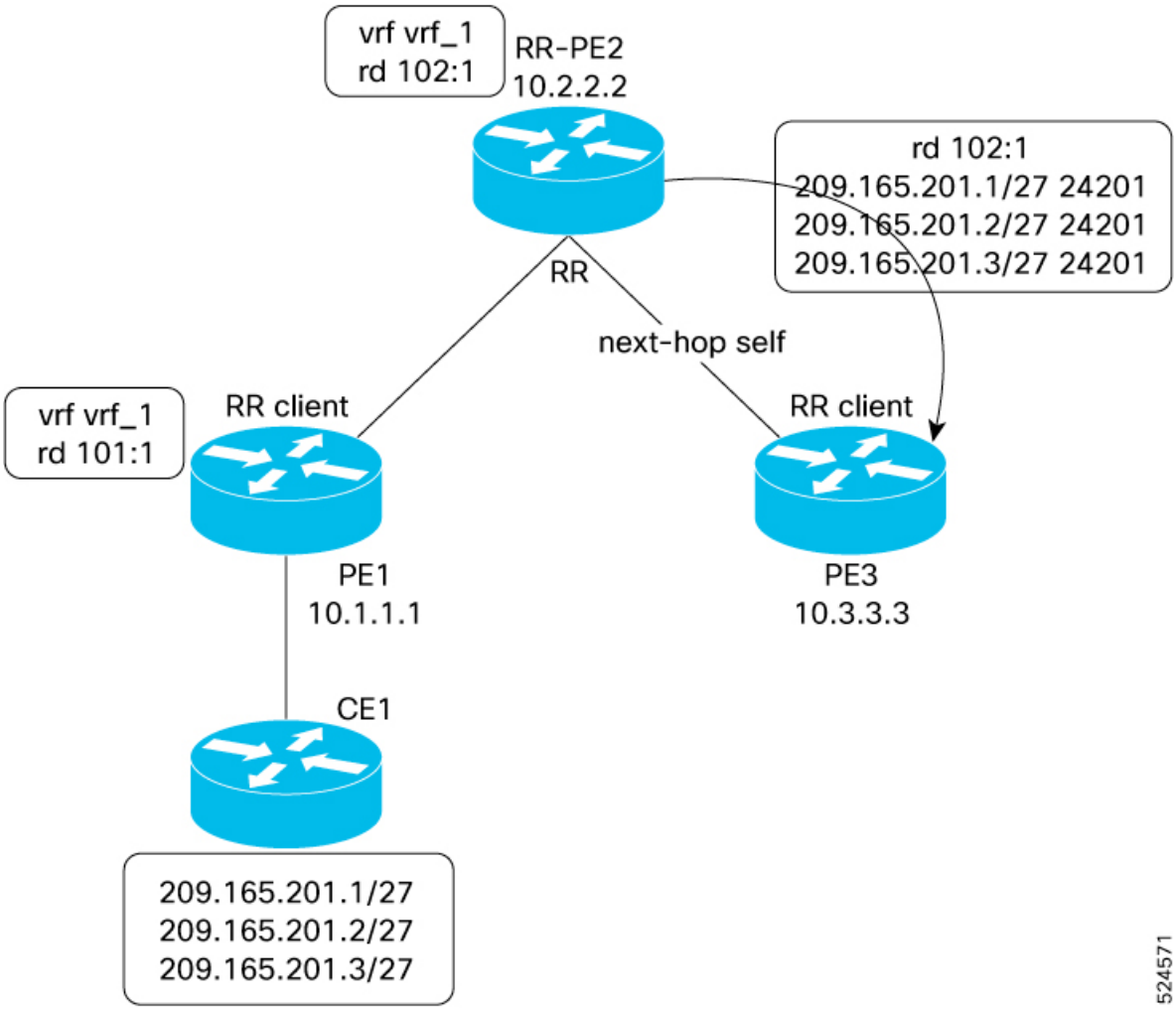
```
Router# show bgp label table
Router# show bgp vpnv4 unicast rd
Router# show mpls label table
Router# show controllers npu resources
```

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 8: 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:

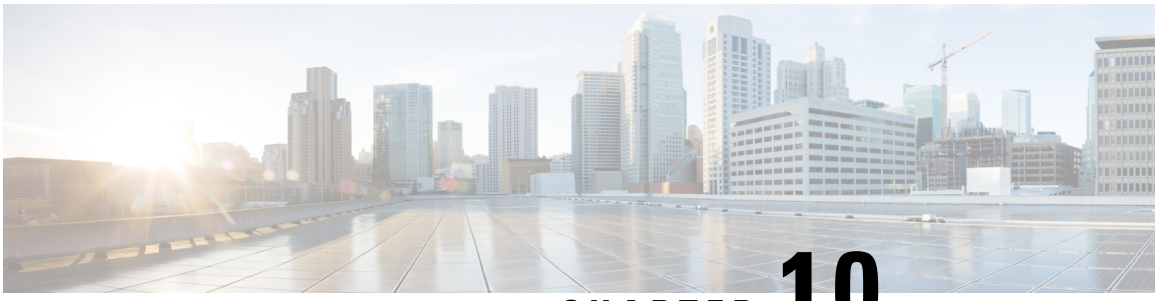
```
Router(config)# route-policy no-set-community
Router(config-rpl)# delete community in com-set-no-advertise
Router(config-rpl)# end-policy
```

```
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
```



CHAPTER 10

BGP Prefix Management and Session Parameters

This chapter covers BGP prefix control, path management, and session-level configurations like TCP MSS customization to optimize network performance.

BGP maximum-prefix and discard extra paths

BGP maximum-prefix is a BGP functionality that enforces a limit on the number of prefixes received from a neighbor for a specific address family and terminates the BGP session if the prefix count exceeds the configured limit.

The discard extra paths option is an enhancement to the BGP maximum-prefix functionality that drops excess prefixes received from a neighbor without flapping the session and limits the memory footprint of BGP.

BGP maximum-prefix overview

The BGP maximum-prefix capability allows you to configure an upper threshold on the number of prefixes that can be accepted from a neighbor for a particular address family. If the number of received prefixes exceeds the configured limit, the system performs these actions:

- Terminates the BGP session and sends a cease notification to the neighbor.
- Keeps the session down until you manually clear it using the **clear bgp** command.
- Restarts the session automatically after a specified period if configured with the **maximum-prefix** command and the **restart** keyword.

Starting with IOS-XR Release 7.3.1, the system no longer applies default limits unless you explicitly configure the maximum number of prefixes for the address family.

Discard extra paths behavior

The discard extra paths option modifies BGP maximum-prefixbehavior. With discard extra paths option configured, when excess prefixes are received, they are dropped, but the BGP session remains stable. If the discard extra paths configuration is removed, BGP sends a route-refresh message to the neighbor if it supports the refresh capability; otherwise, the session flaps.

Effects of changing the maximum-prefix value

Changing the maximum-prefix value triggers these specific actions:

- If the new value exceeds the current prefix count, the additional prefixes are saved.
- If the new value is less than the current prefix count, some prefixes are deleted to align with the new limit. There is no control over which prefixes are removed.

Benefits of BGP maximum-prefix and discard extra paths

BGP maximum-prefix and discard extra paths functionality provides these benefits:

- Limits the memory usage of BGP by dropping excess prefixes received beyond the configured maximum.
- Prevents session flapping, ensuring stability of the BGP peer even when the prefix limit is exceeded.
- Helps maintain operational continuity by avoiding disruptions caused by exceeding the prefix limit.

Limitations of discard extra paths

When configuring discard extra paths, consider these guidelines.

The discard extra paths configuration cannot coexist with the **soft reconfig** configuration.

If the system runs out of physical memory, the BGP process exits and requires a manual restart using the **process restart bpm** command.

- Dropped prefixes can cause network inconsistencies, potentially leading to routing loops.
- During a Non-Stop Routing (NSR) switchover, standby and active BGP sessions may drop different prefixes, causing inconsistent BGP tables.

Benefits of BGP maximum-prefix and discard extra paths

BGP maximum-prefix and discard extra paths functionality provides these benefits:

- Limits the memory usage of BGP by dropping excess prefixes received beyond the configured maximum.
- Prevents session flapping, ensuring stability of the BGP peer even when the prefix limit is exceeded.
- Helps maintain operational continuity by avoiding disruptions caused by exceeding the prefix limit.

Configure BGP maximum-prefix and discard extra paths

The purpose of this task is to configure a BGP neighbor to discard extra paths when the maximum prefix limit is exceeded, ensuring session stability and controlled memory usage.

This task involves setting up a BGP neighbor with a maximum prefix limit and enabling the discard extra paths option to drop prefixes exceeding the limit without session flapping.

Follow these steps to configure BGP maximum-prefix discard extra paths:

Before you begin

- Identify the BGP autonomous system (AS) number.
- Determine the neighbor IP address and the maximum prefix limit to configure.

Procedure

Step 1 Enter configuration mode and specify the BGP autonomous system number to enable BGP configuration.

Example:

```
Router# configure
Router(config)# router bgp 10
router(config-bgp)#
```

Step 2 Define the neighbor IP address and enter the address family submode to configure the specific type of traffic.

Example:

```
Router(config-bgp)# neighbor 10.0.0.1
Router(config-bgp-nbr)# address-family ipv4 unicast
```

Step 3 Configure the maximum prefix limit and enable the discard extra paths option to prevent excess prefixes from causing disruptions.

Example:

```
Router(config-bgp-nbr-af)# maximum-prefix 1000 discard-extra-paths
```

Step 4 Save the configuration to apply changes.

Example:

```
Router(config-bgp-nbr-af)# commit
```

Step 5 Verify the running configuration on the system.

Example:

```
Router# show running-config
router bgp 10
 neighbor 10.0.0.1
 address-family ipv4 unicast
   maximum-prefix 1000 discard-extra-paths
   !
  !
```

Step 6 Use the **show bgp neighbor** command to view the neighbor configuration and status.

Review the output for these details:

- Maximum prefixes allowed
- Discard extra paths configuration
- Threshold for warning messages

Example:

```
Router# show bgp neighbor 10.0.0.1
BGP neighbor is 10.0.0.1
```

```
Maximum prefixes allowed 1000 (discard-extra-paths)
Threshold for warning message 75%
```

The configuration ensures that the BGP neighbor limits the number of prefixes to 1,000 and discards extra paths without flapping the session.

Summary of key commands for BGP maximum-prefix and discard extra paths

Table 13: Key commands

Command	Description
`maximum-prefix <value> discard-extra-paths`	Configures the maximum prefix limit and enables the discard extra paths option.
`show bgp neighbor <neighbor-ip>`	Displays the BGP neighbor's configuration and status, including discard extra paths details.
`process restart bpm`	Manually restarts the BGP process when it exits due to insufficient physical memory.

TCP maximum segment size

Maximum Segment Size (MSS) is a TCP attribute that

- determines the largest amount of data that a device can receive in a single, unfragmented TCP segment
- is limited by the Maximum Transmission Unit (MTU) of an interface, and
- is negotiated during the TCP setup process between a source and destination.

The MSS ensures efficient data transfer by optimizing the size of transmitted packets, especially for protocols like BGP. Each direction of data flow can use a different MSS value based on the MTU of the source and destination interfaces.

Key attributes of MSS

These are some of the key attributes of MSS:

- The closer the MSS is to the MTU, the more efficient the data transfer.
- The MSS is announced during the TCP setup process.
- The default TCP MSS value is 536 octets or 1,460 bytes. This means that TCP segments the data in the transmit queue into 1460-byte chunks before passing the packets to the IP layer.

Per neighbor TCP MSS

Per neighbor TCP MSS is a mechanism in BGP configuration that

- allows creating unique TCP MSS profiles for each neighbor

- supports configuration in two modes: neighbor group and session group, and
- overrides the global TCP MSS setting for specific neighbors.

Key attributes of per neighbor TCP MSS

These are some of the key attributes of per neighbor TCP MSS:

- You can enable or disable TCP MSS configuration for specific neighbors.
- MSS value can be reset to its default using the **inheritance-disable** command.
- The configuration range for MSS values is from 68 to 10,000.

Configure per neighbor TCP MSS

The purpose of this task is to configure a TCP MSS value for a specific neighbor in BGP.

Before you begin

Identify the desired MSS value.

Procedure

Step 1 Enter BGP configuration mode and set up the neighbor group.

Example:

```
Router# configure
Router#(config)# router bgp 10
Router#(config-bgp)# address-family ipv4 unicast
Router#(config-bgp-af)# exit
Router#(config-bgp)# neighbor-group n1
Router#(config-bgp-nbrgrp)# tcp mss 500
Router#(config-bgp-nbrgrp)# address-family ipv4 unicast
Router#(config-bgp-nbrgrp-af)# exit
Router#(config-bgp-nbrgrp)# exit
```

Step 2 Configure a specific neighbor and inherit settings from the neighbor group.

Example:

```
Router#(config-bgp)# neighbor 10.0.0.2
Router#(config-bgp-nbr)# remote-as 1
Router#(config-bgp-nbr)# use neighbor-group n1
Router#(config-bgp-nbr)# address-family ipv4 unicast
Router#(config-bgp-nbr-af)#
```

Step 3 Save the configuration.

Example:

```
Router#(config-bgp-nbr-af)# commit
```

Step 4 Verify the running configuration on the system.

Example:

```
Router# show running-config
router bgp 10
 address-family ipv4 unicast
 !
 neighbor-group n1
  tcp mss 500
  address-family ipv4 unicast
  !
 !
 neighbor 10.0.0.2
  remote-as 1
  use neighbor-group n1
  address-family ipv4 unicast
  !
 !
!
end
```

Step 5 Use the **show bgp neighbor** command to view the neighbor configuration and status.

Example:

```
Router# show bgp neighbor 10.0.0.2
BGP neighbor is 10.0.0.2
 Remote AS 1, local AS 10, external link
 Remote router ID 0.0.0.0
  BGP state = Idle (No best local address found)
  ...
  Minimum time between advertisement runs is 30 secs
  Configured TCP Maximum Segment Size 500
  Inbound message logging enabled, 3 messages buffered
  Outbound message logging enabled, 3 messages buffered
  ...
 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.1 Filter-group: 0.0  No Refresh request being processed
  eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
  ...
```

Step 6 Use the **show tcp brief** command to check TCP connection endpoints.

Example:

```
Router# show tcp brief
   PCB             VRF-ID     Recv-Q Send-Q Local Address          Foreign Address        State
0x000055e27958c800 0x60000000      0      0  :::179                :::0                    LISTEN
0x000055e27958b850 0x00000000      0      0  :::179                :::0                    LISTEN
0x00007f2a80002050 0x60000000      0      0  0.0.0.0:179           0.0.0.0:0               LISTEN
0x00007f2a840380d0 0x00000000      0      0  0.0.0.0:179           0.0.0.0:0               LISTEN
```

This information helps verify the correct configuration and troubleshoot connectivity issues.

Use the **show tcp** command to view detailed TCP connection information.

The TCP MSS is configured for the specified neighbor.

Disable the per neighbor TCP MSS

The purpose of this task is to disable the TCP MSS configuration for a specific neighbor.

Follow these steps to disable the per neighbor TCP MSS:

Before you begin

Ensure the neighbor group or session group is already configured.

Procedure

Step 1 Enter BGP configuration mode.

Example:

```
Router# configure
Router#(config)# router bgp 10
Router#(config-bgp)#
```

Step 2 Disable MSS inheritance for the neighbor group.

Example:

```
Router#(config-bgp)# address-family ipv4 unicast
Router#(config-bgp-af)# exit
Router#(config-bgp)# neighbor-group n1
Router#(config-bgp-nbrgrp)# tcp mss inheritance-disable
Router#(config-bgp-nbrgrp)# address-family ipv4 unicast
Router#(config-bgp-nbrgrp-af)# exit
Router#(config-bgp-nbrgrp)# exit
```

Step 3 Configure a specific neighbor and disable MSS inheritance for the neighbor.

Example:

```
Router#(config-bgp)# neighbor 10.0.0.2
Router#(config-bgp-nbr)# remote-as 1
Router#(config-bgp-nbr)# use neighbor-group n1
Router#(config-bgp-nbr)# tcp mss inheritance-disable
Router#(config-bgp-nbr)# commit
```

Step 4 Verify the running configuration on the system.

Example:

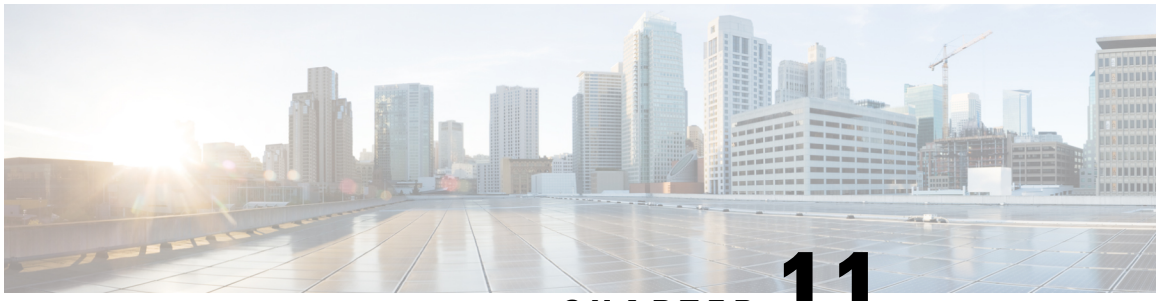
```
Router# show running-config
router bgp 10
 address-family ipv4 unicast
 !
 neighbor-group n1
  tcp mss inheritance-disable
  address-family ipv4 unicast
  !
 !
 neighbor 10.0.0.2
  remote-as 1
  use neighbor-group n1
  tcp mss inheritance-disable
  address-family ipv4 unicast
  !
 !
!
```

TCP MSS is disabled for the specified neighbor.

Summary of key commands for per neighbor TCP MSS

Table 14: Key commands

Command	Description
`show bgp neighbor`	Displays BGP neighbor details, including the configured MSS value.
`show tcp brief`	Lists active TCP connections and their states.
`show tcp pcb <pcb-value>`	Provides detailed TCP connection information for a specific PCB.



CHAPTER 11

BGP Link-State Mechanisms and Update Groups

This chapter covers BGP Link-State fundamentals, configuration tasks for exchanging and distributing IGP link-state information, and update group mechanisms for optimizing BGP update generation and neighbor management.

BGP Link-State

BGP Link-State (BGP-LS) is a protocol extension that

- distributes IGP link-state information via BGP
- improves network topology visibility for applications like SR-PCE, and
- uses a dedicated AFI/SAFI (RFC7752) to encode link-state attributes.

Table 15: Feature History Table

Feature Name	Release Information	Feature Description
BGP Link-State	Release 24.4.1	Introduced in this release on: Fixed Systems(8700)(select variants only*) With BGP Link-State (BGP-LS), you can efficiently share IGP link-state information across your network, allowing applications such as Segment Routing Path Computation Element (SR-PCE) to gain greater topology awareness and optimize path computations using Segment Routing Traffic Engineering (SR-TE). This feature uses Address Family Identifier (AFI) and Sub-address Family Identifier (SAFI) to encode link-state data in the BGP-LS attribute as defined by RFC7752. * BGP-LS functionality is now supported on Cisco-8712-MOD-M routers.

BGP-LS extends BGP to carry Interior Gateway Protocol (IGP) link-state information using a specific Address Family Identifier (AFI) and Sub-address Family Identifier (SAFI). As defined in RFC7752, BGP-LS encodes

each link-state object, such as a node, a link, or a prefix, in the BGP Network Layer Reachability Information (NLRI) format, while the properties of each object are conveyed using the BGP-LS path attribute. This approach enables controllers and applications to build a comprehensive topology view across multiple domains.

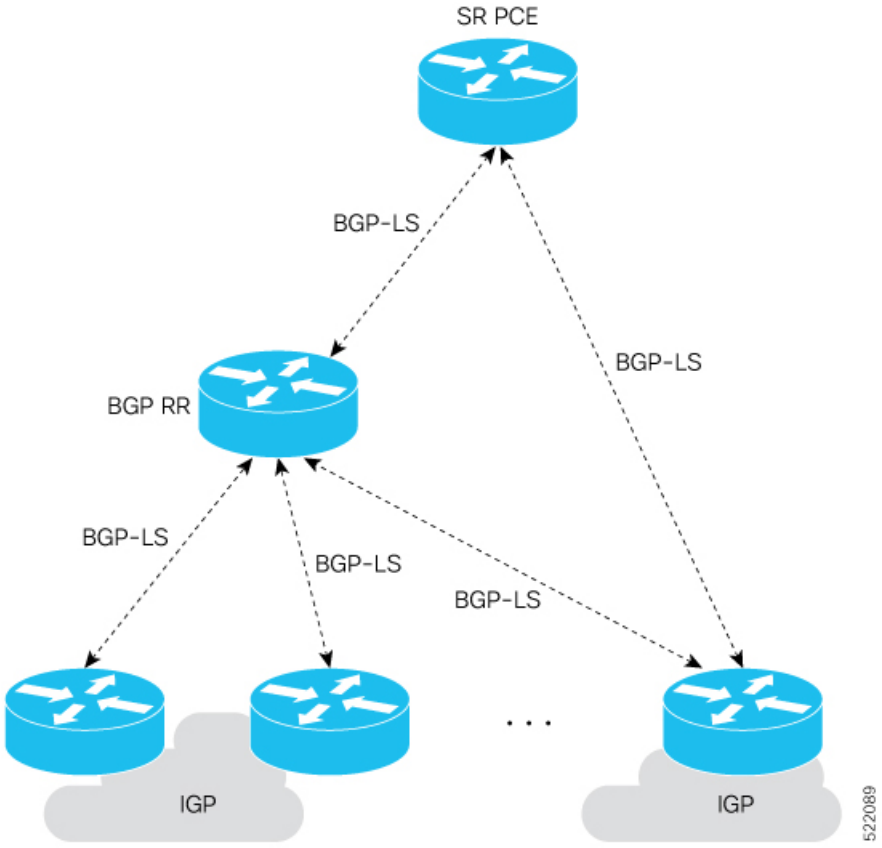
Example

A Segment Routing Path Computation Element (SR-PCE) can use BGP-LS data to discover node capabilities, learn mappings for SR segments, and compute optimal paths using Segment Routing Traffic Engineering (SR-TE). This enables SR-PCE to steer traffic on paths different from the underlying IGP-based distributed best-path computation.

Deployment scenarios for BGP Link-State

This figure illustrates a typical BGP-LS deployment.

Figure 9: Deployment scenario for BGP-LS



The deployment involves:

- Configuring one or more BGP speakers (nodes) with BGP-LS in each IGP area.
- Establishing an iBGP mesh between BGP speakers and route-reflectors.
- Allowing route-reflectors to aggregate and share link-state information from all IGP areas and from eBGP peers in other autonomous systems (AS).

Usage guidelines and limitations for BGP Link-State

Functional limitations

IGPs do not use BGP-LS data from remote BGP peers, and BGP does not download the received BGP-LS information into other router components.

Instance-id usage

The identifier field in BGP Link-State, called the instance-id, specifies the IGP routing domain to which the NLRI is associated.

Some best practices for instance-id configuration are:

- Assign a consistent instance-id value to all BGP-LS producers within a single IGP domain.
- When only one protocol instance is present, configure the instance-id value to 0.
- Use unique instance-id values for different routing protocol instances operating in separate IGP domains.

NLRIs with different instance-id values represent different IGP routing instances. If consistent instance-ids are not used, BGP-LS consumers may see duplicate objects or incorrect topology.

Configure BGP Link-State with a neighbor

Enable the exchange of BGP Link-State information with a BGP neighbor.

Use this task to distribute IGP link-state data (OSPF or IS-IS) to a BGP neighbor for use by controllers or applications such as SR-PCE.

Before you begin

- Ensure you have access to the router CLI.
- Confirm that the neighbor uses a private IP address.

Procedure

Step 1 Run the **configure** comamnd to enter configuration mode.

Example:

```
Router# configure
```

Step 2 Specify the BGP AS number and enter BGP configuration mode.

Example:

```
Router(config)# router bgp 100
```

Step 3 Configure the neighbor using its IP address.

Example:

```
Router(config-bgp)# neighbor 10.0.0.2
```

Step 4 Set the remote AS number for the neighbor and enter the link-state address family.

Example:

```
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# address-family link-state link-state
Router(config-bgp-nbr)# commit
```

The router exchanges BGP Link-State data with the specified neighbor.

Configure a unique domain distinguisher (four-octet ASN)

Assign a unique identifier (domain distinguisher) for BGP Link-State using a four-octet ASN.

The domain distinguisher helps differentiate routing domains when distributing link-state information.

Follow these steps to configure the domain distinguisher.

Procedure

Step 1 In configuration mode, specify the BGP AS number and enter BGP configuration mode.

Example:

```
Router# configure
Router(config)# router bgp 100
```

Step 2 Enter the link-state address family configuration mode.

Example:

```
Router(config-bgp)# address-family link-state link-state
```

Step 3 Assign a unique domain distinguisher.

Example:

```
Router(config-bgp-af)# domain-distinguisher 1234
Router(config-bgp-af)# commit
```

Possible range of domain-distinguisher: 1 to 4294967295

The router uses the specified four-octet ASN as the domain distinguisher for BGP-LS.

Distributing IGP link-state databases with BGP-LS

BGP Link-State (BGP-LS) enables the distribution of IGP link-state databases, including OSPF and IS-IS link-state data, across multiple, independent routing domains. This distribution allows controllers or applications to build end-to-end paths that span several domains.

To distribute OSPFv2 link-state data using BGP-LS:

- Enter router configuration mode.
- Configure the OSPF process.

- Use the **distribute link-state** command with the appropriate instance-id.

Example configuration:

```
Router# configure
Router(config)# router ospf 100
Router(config-ospf)# distribute link-state instance-id 32
```

BGP update groups

A BGP update group is a mechanism that

- groups BGP neighbors with similar outbound routing policies
- enables efficient update message generation, and
- automatically recalculates group membership when configuration changes occur.

BGP update group generation separates update message creation from neighbor configuration by using an algorithm to assign neighbors to groups based on their outbound policies. This process operates automatically without requiring manual configuration. With update group-based message generation, the router sends a single update message to all neighbors in a group, which simplifies management, reduces redundant updates, and improves overall network performance.

How BGP update group membership and optimization work

Summary

The key components involved in the process are:

- Router: Applies outbound policies and manages update groups.
- BGP neighbors: Are assigned to update groups based on their outbound routing policies.
- Update group algorithm: Dynamically calculates group membership and manages update message generation.

BGP update group membership is recalculated automatically whenever there is a configuration change. The router ensures that neighbors with matching outbound routing policies are grouped together, which helps reduce processing and optimize updates.

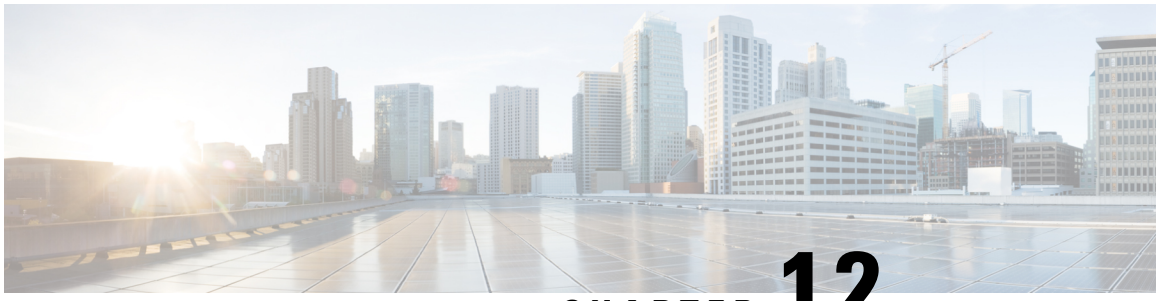
Workflow

These stages describe how BGP update group membership and optimization work.

- The router detects a configuration change.
- The update group algorithm recalculates group memberships based on current outbound policies.
- The router generates update messages for each group and sends them to the appropriate neighbors.

Result

The router delivers optimized BGP updates, reducing redundancy and improving overall network efficiency.



CHAPTER 12

BGP Default Limits and Multiprotocol Address Handling

This chapter explains Border Gateway Protocol (BGP) default limits and multiprotocol address handling, such as advertising IPv4 routes over IPv6 networks. It also covers how to configure a Multicast Distribution Tree (MDT) address family session in BGP, with examples for default and non-default VRFs.

BGP default peer and prefix limits

BGP default limits are safeguards that

- restrict the maximum number of BGP peers and prefixes to prevent resource depletion,
- allow network operators to control neighbor and prefix scaling, and
- help maintain router stability in case of misconfiguration.

Table 16: Feature History Table

Feature Name	Release Information	Feature Description
Support for increased number of BGP peers	Release 7.3.1	This feature is now enhanced to support 750 IPv4 and 750 IPv6 BGP peers.

BGP imposes maximum limits on how many neighbors you can configure on a router and on the number of prefixes the router accepts from each peer for each address family.

Default BGP peer and prefix limit values

These default limits apply to BGP configurations:

- Maximum number of BGP peers:
  - Default: 4000

- Configurable range: 1 to 15000

  Use the **bgp maximum neighbor** command to change.

- Attempts to exceed the maximum or reduce it below the current number of peers will fail.

- ***Table 17: Default maximum prefixes accepted per peer, by address family***

Address family	Default maximum prefixes
IPv4 unicast	1,048,576
IPv4 labeled-unicast	131,072
IPv4 tunnel	1,048,576
IPv6 unicast	524,288
IPv6 labeled-unicast	131,072
IPv4 multicast	131,072
IPv6 multicast	131,072
IPv4 MVPN	2,097,152
VPNv4 unicast	2,097,152
IPv4 MDT	131,072
VPNv6 unicast	1,048,576
L2VPN EVPN	2,097,152

You can override these values using the **maximum-prefix limit** command for each peer and address family.

How BGP default limits work

Summary

The key components involved in the process are:

- Router: Tracks peer and prefix counts.
- BGP neighbor: Sends prefix advertisements.
- Configuration commands: Allow you to adjust limits as needed.

BGP monitors the number of peers and received prefixes to enforce default or configured limits, protecting the router from excessive resource usage.

Workflow

These stages describe how BGP default limits work:

- The router checks the number of configured peers when changes are made.

- If a peer tries to exceed the maximum, the router rejects the configuration.
- For prefixes, the router monitors the count for each neighbor and address family.
- If the number of received prefixes exceeds the limit, either the default or the configured limit, the router sends a cease notification and terminates the BGP peering with that neighbor.
- If you lower the maximum-prefix limit after the session has started and the neighbor has already sent more prefixes than allowed, the router immediately sends a cease notification and ends the peering.

Result: BGP protects router resources and ensures reliable operation by enforcing these limits and terminating peerings that exceed them.

IPv4 NLRI advertisement with IPv6 next hops in multiprotocol BGP networks

Advertising IPv4 network layer reachability information (NLRI) with IPv6 next hops is a multiprotocol BGP feature that

- enables seamless communication between IPv4 devices across an IPv6 core
- supports both default and non-default virtual routing and forwarding (VRF) instances, and
- increases network flexibility and efficiency by allowing IPv4 routes to use IPv6 next hops without requiring address family alignment between network segments.

Table 18: Feature History Table

Feature Name	Release Information	Feature Description
Advertising IPv4 NLRI with IPv6 next hops in the non-default VRF	Release 24.4.1	Introduced in this release on: Fixed Systems(8700)(select variants only*) *Advertising IPv4 NLRI with IPv6 NextHops in the non-default VRF is now supported on Cisco 8712-MOD-M routers
Advertising IPv4 NLRI with IPv6 next hops in the non-default VRF	Release 24.2.11	This feature enhances network efficiency and security by allowing you to create default and non-default virtual routing tables. Thesetables isolate traffic through customized routing policies, allowing for the communication of IPv4 address family over IPv6 next hops specifically within non-default VRFs.

Advertising IPv4 NLRI with IPv6 next hops in MP-BGP networks	Release 7.3.1	With the capability of Multiprotocol BGP for exchanging IPv4 address family over IPv6 next hop, legacy IPv4 packets from edge devices can traverse through IPv6 core networks seamlessly. This feature allows IPv4 NLRI to be encoded and advertised over IPv6 next hop in a single BGP session. This feature supports exchanging the IPv4 address family over the IPv6 next hop only in the default VRF. Configure the **ipv4 forwarding-enable** command on the edge router connected to the core interface to encode IPv4 packets over IPv6 next hop.

You can use this feature in networks where devices with different address families coexist. For example, an IPv4 core may be surrounded by IPv6 devices, or the reverse. In these mixed environments, multiprotocol BGP (MP-BGP) allows you to advertise IPv4 routes using IPv6 next hops, so reachability information can cross address family boundaries. Common reasons for deploying such topologies include limited availability of IPv4 addresses for interface assignment and phased migration to an all-IPv6 network.

By enabling IPv4 forwarding on interfaces, you ensure that the type of address family used does not limit traffic flow across the network.

How IPv4 NLRI advertisement with IPv6 next hops works

Summary

The key components involved in the process are:

- BGP speakers: Routers that participate in multiprotocol BGP and advertise NLRI.
- VRFs: Routing tables that isolate traffic and allow for customized policy application.
- Next hop addresses: IPv6 addresses used as the next hop for IPv4 NLRI.

The router uses multiprotocol BGP to encode and advertise IPv4 routes with IPv6 next hops, supporting both default and non-default VRF instances. This allows legacy IPv4 packets to traverse an IPv6 network core.

Workflow

These stages describe how IPv4 NLRI advertisement with IPv6 next hops works:

- Configure interfaces with IPv6 addresses and enable IPv4 forwarding.
- Create route policies to set IPv6 next hops for IPv4 prefixes.
- Establish BGP neighbor relationships using IPv6 addresses.
- Advertise IPv4 NLRI with IPv6 next hop information through BGP sessions.
- Verify advertisement and routing through show commands.

Result: IPv4 routes are advertised and reachable over IPv6 next hops, supporting seamless traffic flow in mixed-protocol environments.

Configure IPv4 NLRI advertisement with IPv6 next hops in the default VRF

Enable advertisement of IPv4 NLRI with IPv6 next hops in the default VRF.

In the following example, the eBGP peer is configured on the HundredGigE 0/0/0/24 interface, and the iBGP peer is configured on the HundredGigE 0/0/0/25 interface of the router being configured.

Procedure

Step 1 Configure the required interfaces with IPv6 addresses and enable IPv4 forwarding.

Example:

```
Router# configure
Router(config)# interface HundredGigE 0/0/0/24
Router(config-if)# ipv6 address 2000::2/64
Router(config-if)# ipv4 forwarding-enable
Router(config-if)# no shutdown
Router(config-if)# commit
Router(config-if)# exit

Router(config)# interface HundredGigE 0/0/0/25
Router(config-if)# ipv6 address 3000::2/64
Router(config-if)# ipv4 forwarding-enable
Router(config-if)# no shutdown
Router(config-if)# commit
Router(config-if)# exit
```

Step 2 Create a prefix set and route policy to set IPv6 next hop for IPv4 prefixes.

Example:

```
Router(config)# route-policy 5549
Router(config-rpl)# if destination in 5549-pfx then set next-hop 20:20::20:200 endif
Router(config-rpl)# end-policy
Router(config)# prefix-set 5549-pfx
Router(config-pfx)# 3.3.3.3/32, 100.1.1.0/24 end-set
Router(config)# commit
```

Step 3 Configure iBGP and eBGP peers using IPv6 addresses and apply the route policies.

Example:

```
Router(config)# route-policy pass-all
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# address-family ipv4 multicast
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 3000::1
Router(config-bgp-nbr)# remote-as 100 Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# next-hop-self Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# commit
Router(config-bgp-nbr-af)# root
```

Step 4 Configure an eBGP peer through the HundredGigE 0/0/0/24 interface and use the route policy for setting an IPv6 nexthop for IPv4 routes.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# address-family ipv4 multicast
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 2000::1
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy 5549 out
Router(config-bgp-nbr-af)# commit
Router(config-bgp-nbr-af)# root
```

Step 5 Verify the running configuration.

Example:

```
Router# show running-config
...
interface HundredGigE0/0/0/24 ipv4 forwarding-enable
ipv6 address 2000::2/64
!
interface HundredGigE0/0/0/25 ipv4 forwarding-enable
ipv6 address 3000::2/64
!
prefix-set 5549-pfx 3.3.3.3/32,
100.1.1.0/24
end-set
!
route-policy 5549
if destination in 5549-pfx then set next-hop 20:20::20:200
endif end-policy
!
route-policy pass-all pass
end-policy
!
router bgp 100
address-family ipv4 unicast
!
address-family ipv4 multicast
!
neighbor 2000::1
remote-as 200
address-family ipv4 unicast route-policy pass-all in route-policy 5549 out
!
!
neighbor 3000::1
remote-as 100
address-family ipv4 unicast route-policy pass-all in route-policy pass-all out next-hop-self
!
!
!
end
```

Step 6 Use these show commands to verify the BGP neighbor configuration and the advertisement of next hops.

Example:

```
Router# show bgp neighbor
BGP neighbor is 10:10::10:10
Remote AS 100, local AS 100, internal link Remote router ID 10.10.10.10
Cluster ID 30.30.30.30
```

```
BGP state = Established, up for 11:42:17 NSR State: NSR Ready
Last read 00:00:10, Last read before reset 00:00:00 Hold time is 180, keepalive interval is 60 seconds
Configured hold time: 180, keepalive: 60, min acceptable hold time: 3 Last write 00:00:09, attempted
 19, written 19
Second last write 00:01:09, attempted 19, written 19
Last write before reset 00:00:00, attempted 0, written 0 Second last write before reset 00:00:00,
attempted 0, written 0
Last write pulse rcvd Jun 12 11:57:40.005 last full Jun 12 03:38:09.496 pulse count 1766

Last write pulse rcvd before reset 00:00:00
Socket not armed for io, armed for read, armed for write
Last write thread event before reset 00:00:00, second last 00:00:00 Last KA expiry before reset
00:00:00, second last 00:00:00
Last KA error before reset 00:00:00, KA not sent 00:00:00 Last KA start before reset 00:00:00, second
 last 00:00:00 Precedence: internet
Non-stop routing is enabled
Multi-protocol capability received Neighbor capabilities:
Route refresh: advertised (old + new) and received (old + new) 4-byte AS: advertised and received
Address family IPv4 Unicast: advertised and received Received 866 messages, 0 notifications, 0 in
queue
Sent 1021 messages, 0 notifications, 0 in queue Minimum time between advertisement runs is 0 secs
Inbound message logging enabled, 3 messages buffered Outbound message logging enabled, 3 messages
buffered

For Address Family: IPv4 Unicast BGP neighbor version 120021
Update group: 0.2 Filter-group: 0.2 No Refresh request being processed Route-Reflector Client
Extended Nexthop Encoding: advertised and received
Route refresh request: received 0, sent 0
11 accepted prefixes, 11 are bestpaths Exact no. of prefixes denied : 0. Cumulative no. of prefixes
 denied: 0.
Prefix advertised 60006, suppressed 0, withdrawn 60000 Maximum prefixes allowed 1048576
Threshold for warning message 75%, restart interval 0 min AIGP is enabled
An EoR was not received during read-only mode
Last ack version 120021, Last synced ack version 120021 Outstanding version objects: current 0, max
 3 Additional-paths operation: None
Send Multicast Attributes
Advertise routes with local-label via Unicast SAFI

Connections established 1; dropped 0
Local host: 30:30::30:30, Local port: 51453, IF Handle: 0x00000000 Foreign host: 10:10::10:10, Foreign
 port: 179
Last reset 00:00:00


Router# show bgp ipv4 unicast update-group
Mon Jun 12 11:47:31.543 UTC

Update group for IPv4 Unicast, index 0.2: Attributes:
Neighbor sessions are IPv6 Internal
Common admin
First neighbor AS: 100 Send communities
Send GSHUT community if originated Send extended communities
Route Reflector Client 4-byte AS capable
Advertise routes with local-label via Unicast SAFI Send AIGP
Send multicast attributes Extended Nexthop Encoding
Minimum advertisement interval: 0 secs Update group desynchronized: 0
Sub-groups merged: 5
Number of refresh subgroups: 0
Messages formatted: 156, replicated: 228 All neighbors are assigned to sub-group(s)
Neighbors in sub-group: 0.2, Filter-Groups num:1 Neighbors in filter-group: 0.2(RT num: 0)
10:10::10:10
20:20::20:20


Router# show bgp 3.3.3.3/32
```

```
Mon Jun 12 11:57:59.451 UTC
BGP routing table entry for 3.3.3.3/32 Versions:
Process bRIB/RIB SendTblVer
Speaker 21 21
Last Modified: Jun 12 00:15:45.314 for 11:42:14
Paths: (1 available, best #1)
Advertised to update-groups (with more than one peer): 0.2
Path #1: Received by speaker 0
Advertised to update-groups (with more than one peer): 0.2
3000, (Received from a RR-client)
20:20::20:20 (metric 1) from 20:20::20:20 (20.20.20.20)
Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best Received Path ID 0,
Local Path ID 0, version 21
```

```
Router# show bgp ipv4 unicast nexthops
...
…….. Snippet …..

Gateway Address Family: IPv6 Unicast Table ID: 0xe0800000
Nexthop Count: 2
Critical Trigger Delay: 3000msec
Non-critical Trigger Delay: 10000msec

Nexthop Version: 3, RIB version: 1
EPE Table Version: 1, EPE Label version: 1
EPE Downloaded Version: 1, EPE Standby Version: 1

Status codes: R/UR Reachable/Unreachable
C/NC Connected/Not-connected L/NL Local/Non-local
PR Pending Registration I Invalid (Policy drop)
Next Hop Status Metric Tbl-ID Notf LastRIBEvent RefCount
10:10::10:10 [R][NC][NL] 1e0800000 1/0 11:42:43 (Cri) 11/14
20:20::20:20 [R][NC][NL] 1e0800000 1/0 11:42:43 (Cri) 8/11
```

```
Router# show bgp ipv4 unicast nexthops 10:10::10:10
Nexthop: 10:10::10:10 VRF: Default
Nexthop ID: 0x6000001, Version: 0x2 Nexthop Flags: 0x00000080
Nexthop Handle: 0x7f53e85b136c

RIB Related Information:
Firsthop interface handle 0x00000140
Gateway TBL Id: 0xe0800000 Gateway Flags: 0x00000080 Gateway Handle: 0x7f540d2e8f00
Gateway: reachable, non-Connected route, prefix length 128 Resolving Route: 10:10::10:10/128 (ospf
100)
Paths: 0
RIB Nexhop ID: 0x0
Status: [Reachable][Not Connected][Not Local] Metric: 1
Registration: Asynchronous, Completed: 2d21h Events: Critical (1)/Non-critical (0)
Last Received: 11:42:55 (Critical)
Last gw update: (Crit-notif) 11:42:55(rib) Reference Count: 11

Prefix Related Information Active Tables: [IPv4 Unicast] Metrices: [0x1]
Reference Counts: [11] Interface Handle: 0x0
```

```
Router# show route 3.3.3.3/32
Mon Jun 12 12:32:13.503 UTC

Routing entry for 3.3.3.3/32
Known via "bgp 100", distance 200, metric 0 Tag 3000, type internal
Installed Jun 12 00:15:45.626 for 12:16:28 Routing Descriptor Blocks
20:20::20:20, from 20:20::20:20
Route metric is 0 No advertising protos.
```

```
Router# show route 3.3.3.3/32 detail
Mon Jun 12 12:32:16.447 UTC
```

```
Routing entry for 3.3.3.3/32
Known via "bgp 100", distance 200, metric 0 Tag 3000, type internal
Installed Jun 12 00:15:45.628 for 12:16:30 Routing Descriptor Blocks
20:20::20:20, from 20:20::20:20
Route metric is 0 Label: None Tunnel ID: None
Binding Label: None
Extended communities count: 0 NHID:0x0(Ref:0)
Route version is 0x1 (1) No local label
IP Precedence: Not Set QoS Group ID: Not Set Flow-tag: Not Set
Fwd-class: Not Set
Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_LOCAL
Download Priority 4, Download Version 24 No advertising protos.


Router# show cef 3.3.3.3/32
Mon Jun 12 12:32:22.627 UTC
3.3.3.3/32, version 24, internal 0x5000001 0x0 (ptr 0x8e0b76b8) [1], 0x0 (0x0), 0x0 (0x0)
Updated Jun 12 00:15:45.631
local adjacency fe80::f83b:74ff:fe65:f004
Prefix Len 32, traffic index 0, precedence n/a, priority 4
via 20:20::20:20/128, 2 dependencies, recursive [flags 0x6000] path-idx 0 NHID 0x0 [0x8e352034 0x0]
next hop VRF - 'default', table - 0xe0800000 next hop 20:20::20:20/128 via 20:20::20:20/128


Router# show cef 3.3.3.3/32 detail
Mon Jun 12 12:32:25.415 UTC
3.3.3.3/32, version 24, internal 0x5000001 0x0 (ptr 0x8e0b76b8) [1], 0x0 (0x0), 0x0 (0x0)
Updated Jun 12 00:15:45.632
local adjacency fe80::f83b:74ff:fe65:f004
Prefix Len 32, traffic index 0, precedence n/a, priority 4
gateway array (0x8eec3170) reference count 6, flags 0x2010, source rib (7), 0 backups [1 type 3 flags
 0x48501 (0x8e191998) ext 0x0 (0x0)]
LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
gateway array update type-time 1 Jun 12 00:15:45.632
LDI Update time Jun 12 00:15:45.632
via 20:20::20:20/128, 2 dependencies, recursive [flags 0x6000] path-idx 0 NHID 0x0 [0x8e352034 0x0]
next hop VRF - 'default', table - 0xe0800000 next hop 20:20::20:20/128 via 20:20::20:20/128

Load distribution: 0 1 (refcount 1)

Hash OK Interface Address
0 Y FortyGigE0/0/0/0 fe80::f83b:74ff:fe65:f004
1 Y FortyGigE0/0/0/25 fe80::f83b:74ff:fe65:f08c
```

You have successfully configured and verified the advertisement of IPv4 NLRI through IPv6 nexthops in the default VRF.

Configure IPv4 NLRI advertisement with IPv6 next hops in a non-default VRF

Enable advertisement of IPv4 NLRI with IPv6 next hops in a non-default VRF.

Procedure

Step 1 Create and configure the VRF.

Example:

```
Router# configure
Router(config)# vrf RFC5549
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# exit
Router(config-vrf)# address-family ipv4 multicast
Router(config-vrf-af)# exit
Router(config-vrf)# commit
```

Step 2 Configure the required interfaces with an IPv6 address to the eBGP neighbor and the **ipv4 forwarding-enable** command.

Example:

```
Router# configure
Router(config)# interface GigabitEthernet0/1/0/0.2
Router(config-subif)# vrf RFC5549
Router(config-subif)# ipv4 forwarding-enable
Router(config-subif)# ipv6 address 2001:DB8:2::2/64
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# commit
```

Step 3 Configure the required interfaces with an IPv6 address to the iBGP neighbor and the **ipv4 forwarding-enable** command.

Example:

```
Router# configure
Router(config)# interface GigabitEthernet0/1/0/1.2
Router(config-subif)# vrf RFC5549
Router(config-subif)# ipv4 forwarding-enable
Router(config-subif)# ipv6 address 2001:DB9:2::2/64
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# commit
```

Step 4 Create a route policy to pass all.

Example:

```
Router# configure
Router(config)# route-policy pass-all
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config)# commit
```

Step 5 Configure BGP, VRF, and the required address families.

Example:

```
Router# configure
Router(config)# router bgp 2
Router(config-bgp)# bgp router-id 192.0.2.2
Router(config-bgp)# address-family vpnv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# address-family vpnv4 multicast
Router(config-bgp-af)# exit
Router(config-bgp)# vrf RFC5549
Router(config-bgp-vrf)# rd 1:100
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# exit
Router(config-bgp-vrf)# address-family ipv4 multicast
Router(config-bgp-vrf-af)# exit
Router(config-bgp-vrf)# commit
```

Step 6 Configure the eBGP neighbor.

Example:

```
Router# configure
Router(config)# router bgp 2
Router(config-bgp)# vrf RFC5549
Router(config-bgp-vrf)# neighbor 2001:DB8:2::1
Router(config-bgp-vrf-nbr)# remote-as 1
Router(config-bgp-vrf-nbr)# description eBGP neighbor in VRF
Router(config-bgp-vrf-nbr)# address-family ipv4 unicast
Router(config-bgp-vrf-nbr-af)# route-policy pass-all in
Router(config-bgp-vrf-nbr-af)# route-policy pass-all out
Router(config-bgp-vrf-nbr-af)# commit
```

Step 7 Configure the iBGP neighbor.

Example:

```
Router# configure
Router(config)# router bgp 2
Router(config-bgp)# vrf RFC5549
Router(config-bgp-vrf)# neighbor 2001:DB9:2::3
Router(config-bgp-vrf-nbr)# remote-as 2
Router(config-bgp-vrf-nbr)# description iBGP neighbor in VRF
Router(config-bgp-vrf-nbr)# address-family ipv4 unicast
Router(config-bgp-vrf-nbr-af)# route-policy pass-all in
Router(config-bgp-vrf-nbr-af)# route-policy pass-all out
Router(config-bgp-vrf-nbr-af)# commit
```

Verify IPv4 NLRI advertisement with IPv6 next hops in a non-default VRF

Verify advertisement of IPv4 NLRI with IPv6 next hops in a non-default VRF.

Procedure

Step 1 Verify the eBGP and iBGP VRF neighbors.

Example:

```
Router# show bgp vrf RFC5549 neighbors BGP neighbor is 2001:DB8:2::1, vrf RFC5549
Remote AS 1, local AS 2, external link Remote router ID 192.0.2.1
BGP state = Established, up for 00:03:37
Previous State: Idle
Last Received Message: KeepAlive NSR State: None
Last read 00:00:32, Last read before reset 00:05:07 Hold time is 180, keepalive interval is 60 seconds
Configured hold time: 180, keepalive: 60, min acceptable hold time: 3 Last write 00:00:32, attempted
 19, written 19
Second last write 00:01:32, attempted 19, written 19
Last write before reset 00:05:06, attempted 19, written 19 Second last write before reset 00:06:06,
 attempted 19, written 19
Last write pulse rcvd Mar 27 23:40:00.449 last full not set pulse count 79 Last write pulse rcvd
before reset 00:05:06
Last insert into reset queue: Mar 27 23:36:20.435, removed at Mar 27 23:36:20.435 Socket not armed
for io, armed for read, armed for write
Last write thread event before reset 00:04:12, second last 00:05:06 Last KA expiry before reset
00:05:06, second last 00:06:06
Last KA error before reset 00:00:00, KA not sent 00:00:00 Last KA start before reset 00:05:06, second
 last 00:06:06 Precedence: internet
Non-stop routing is enabled Enforcing first AS is enabled Multi-protocol capability received Neighbor
 capabilities:
```

```
Route refresh: advertised (old + new) and received (old + new) 4-byte AS: advertised and received
Address family IPv4 Unicast: advertised and received Received 42 messages, 0 notifications, 0 in
queue
Sent 42 messages, 1 notifications, 0 in queue Minimum time between advertisement runs is 0 secs
Inbound message logging enabled, 3 messages buffered Outbound message logging enabled, 3 messages
buffered Fast fallover is enabled
Neighbor is directly connected
Neighbor fast-fallover is not configured
Neighbor is external and fast-external-fallover is not disabled

For Address Family: IPv4 Unicast BGP neighbor version 10
Update group: 0.3 Filter-group: 0.4 No Refresh request being processed AF-dependent capabilities:
Extended Nexthop Encoding: advertised and received
Route refresh request: received 0, sent 0 Policy for incoming advertisements is pass-all Policy for
 outgoing advertisements is pass-all
1 accepted prefixes, 1 are bestpaths Exact no. of prefixes denied: 0 Cumulative no. of prefixes
denied: 0
Prefix advertised 1, suppressed 0, withdrawn 0 An EoR was received during read-only mode Last ack
version 10, Last synced ack version 0
Outstanding version objects: current 0, max 1, refresh 0 Additional-paths operation: None
Advertise routes with local-label via Unicast SAFI Slow Peer State: Detection-only
Detected state: FALSE, Detection threshold: 300 Detection Count: 0, Recovery Count: 0

Connections established 2; dropped 1
Local host: 2001:DB8:2::2, Local port: 18907, IF Handle: 0x008001e0 Foreign host: 2001:DB8:2::1,
Foreign port: 179
Last reset 00:04:12, due to Address family removed (CEASE notification sent - configuration change)
Time since last notification sent to neighbor: 00:04:12 Error Code: configuration change
Notification data sent:
None

BGP neighbor is 2001:DB9:2::3, vrf RFC5549 Remote AS 2, local AS 2, internal link Remote router ID
192.0.2.3
BGP state = Established, up for 00:03:50
Previous State: Idle
Last Received Message: KeepAlive NSR State: None
Last read 00:00:45, Last read before reset 00:05:06 Hold time is 180, keepalive interval is 60 seconds
Configured hold time: 180, keepalive: 60, min acceptable hold time: 3 Last write 00:00:32, attempted
 19, written 19
Second last write 00:01:32, attempted 19, written 19
Last write before reset 00:05:06, attempted 19, written 19 Second last write before reset 00:06:06,
 attempted 19, written 19
Last write pulse rcvd Mar 27 23:40:00.459 last full not set pulse count 80 Last write pulse rcvd
before reset 00:05:06
Last insert into reset queue: Mar 27 23:36:20.434, removed at Mar 27 23:36:20.434 Socket not armed
for io, armed for read, armed for write
Last write thread event before reset 00:05:06, second last 00:05:06 Last KA expiry before reset
00:05:06, second last 00:06:06
Last KA error before reset 00:00:00, KA not sent 00:00:00 Last KA start before reset 00:05:06, second
 last 00:06:06 Precedence: internet
Non-stop routing is enabled
Multi-protocol capability received Neighbor capabilities:
Route refresh: advertised (old + new) and received (old + new) 4-byte AS: advertised and received
Address family IPv4 Unicast: advertised and received Received 42 messages, 0 notifications, 0 in
queue
Sent 44 messages, 1 notifications, 0 in queue Minimum time between advertisement runs is 0 secs
Inbound message logging enabled, 3 messages buffered Outbound message logging enabled, 3 messages
buffered Fast fallover is not enabled
Neighbor is directly connected
Neighbor fast-fallover is not configured

For Address Family: IPv4 Unicast BGP neighbor version 10
Update group: 0.1 Filter-group: 0.3 No Refresh request being processed NEXT_HOP is always this router
AF-dependent capabilities:
```

```
Extended Nexthop Encoding: advertised and received
Route refresh request: received 0, sent 0 Policy for incoming advertisements is pass-all Policy for
 outgoing advertisements is pass-all
1 accepted prefixes, 1 are bestpaths Exact no. of prefixes denied: 0 Cumulative no. of prefixes
denied: 0
Prefix advertised 1, suppressed 0, withdrawn 0 An EoR was received during read-only mode Last ack
version 10, Last synced ack version 0
Outstanding version objects: current 0, max 1, refresh 0 Additional-paths operation: None
Send Multicast Attributes
Advertise routes with local-label via Unicast SAFI Slow Peer State: Detection-only
Detected state: FALSE, Detection threshold: 300 Detection Count: 0, Recovery Count: 0
Connections established 2; dropped 1
Local host: 2001:DB9:2::2, Local port: 179, IF Handle: 0x00800240 Foreign host: 2001:DB9:2::3, Foreign
 port: 19682
Last reset 00:04:12, due to Address family removed (CEASE notification sent - configuration change)
Time since last notification sent to neighbor: 00:04:12 Error Code: configuration change
Notification data sent:
None
```

Step 2 Verify update groups for the eBGP and iBGP neighbors.

Example:

```
Router# show bgp vrf RFC5549 ipv4 unicast update-group
Update group for IPv4 Unicast, index 0.1:
Attributes:
Neighbor sessions are IPv6 Outbound policy: pass-all Internal
Common admin
First neighbor AS: 2
Send communities
Send GSHUT community if originated Send extended communities
Next-hop-self enabled 4-byte AS capable
Advertise routes with local-label via Unicast SAFI Send multicast attributes
Extended Nexthop Encoding
Minimum advertisement interval: 0 secs Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0 Messages formatted: 2, replicated: 2
All neighbor are assigned to sub-group(s)
Neighbors in sub-group: 0.1, Filter-Groups num:1 Neighbors in filter-group: 0.3(RT num: 0)
2001:DB9:2::3

Update group for IPv4 Unicast, index 0.3: Attributes:
Outbound policy: pass-all First neighbor AS: 1 Directly connected IPv6 EBGP 4-byte AS capable
Advertise routes with local-label via Unicast SAFI
Extended Nexthop Encoding
Minimum advertisement interval: 0 secs Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0 Messages formatted: 2, replicated: 2
All neighbor are assigned to sub-group(s)
Neighbors in sub-group: 0.2, Filter-Groups num:1 Neighbors in filter-group: 0.4(RT num: 0)
2001:DB8:2::1
```

Step 3 Verify eBGP and iBGP IPv4 routes with IPv6 nexthops.

Example:

```
Router# show bgp vrf RFC5549 ipv4 unicast
Wed Mar 27 23:41:44.494 IST
BGP VRF RFC5549, state: Active
BGP Route Distinguisher: 1:100
VRF ID: 0x60000003
BGP router identifier 100.1.1.2, local AS number 2
Non-stop routing is enabled BGP table state: Active
Table ID: 0xe0000012 RD version: 10 BGP table nexthop route policy:
```

```
BGP main routing table version 10
BGP NSR Initial initsync version 6 (Reached) BGP NSR/ISSU Sync-Group versions 0/0

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 Route Distinguisher: 1:100 (default for vrf RFC5549)
Route Distinguisher Version: 10
*> 192.1.0.1/32 2001:DB8:2::1 0 0 1 ? N
*>i192.1.0.3/32 2001:DB9:2::3 0 100 0 ? N

Processed 2 prefixes, 2 paths
```

Step 4 Verify eBGP and iBGP routes.

a) Verify eBGP route.

Example:

```
Router# show bgp vrf RFC5549 ipv4 unicast 192.1.0.1/32
BGP routing table entry for 192.1.0.1/32, Route Distinguisher: 1:100
Versions:
Process bRIB/RIB SendTblVer
Speaker 10 10
Local Label: 24004
Last Modified: Mar 27 23:37:00.000 for 00:04:54 Last Delayed at: ---
Paths: (1 available, best #1)
Advertised to CE peers (in unique update groups):
2001:DB9:2::3
Path #1: Received by speaker 0
Advertised to CE peers (in unique update groups):
2001:DB9:2::3
2001:DB8:2::1 from 2001:DB8:2::1 (192.0.2.1)
Origin incomplete, metric 0, localpref 100, valid, external, best, group-best, import-candidate
Received Path ID 0, Local Path ID 1, version 10 Origin-AS validity: (disabled)
```

b) Verify iBGP route.

Example:

```
Router# show bgp vrf RFC5549 ipv4 unicast 192.1. 0.3/32
BGP routing table entry for 192.1.0.3/32, Route Distinguisher: 1:100
Versions:
Process bRIB/RIB SendTblVer
Speaker 9 9
Local Label: 24007
Last Modified: Mar 27 23:36:47.000 for 00:05:29 Last Delayed at: ---
Paths: (1 available, best #1)
Advertised to CE peers (in unique update groups):
2001:DB8:2::1
Path #1: Received by speaker 0
Advertised to CE peers (in unique update groups): 2001:DB8:2::1
Local
2001:DB9:2::3 from 2001:DB9:2::3 (192.0.2.3)
Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best, import-candidate
Received Path ID 0, Local Path ID 1, version 9
```

Step 5 Display IPv4 and IPv6 next-hop addresses for eBGP and iBGP IPv4 routes in the VRF.

Example:

```
Router# show bgp vrf RFC5549 ipv4 unicast nexthops
Total Nexthop Processing Time Spent: 0.000 secs

Maximum Nexthop Processing Received: 00:00:00
Bestpaths Deleted: 0
Bestpaths Changed: 0 Time Spent: 0.000 secs
```

```
Last Notification Processing Received: 00:37:03
Time Spent: 0.000 secs IPv4 Unicast is active
Gateway Address Family: IPv4 Unicast
Table ID: 0xe0000012
Gateway Reference Count: 1 Gateway AF Bits : 0x1 Nexthop Count: 0
Critical Trigger Delay: 50msec
Non-critical Trigger Delay: 10000msec

Nexthop Version: 1, RIB version: 1
EPE Table Version: 1, EPE Label version: 0
EPE Downloaded Version: 0, EPE Standby Version: 0 IPv4 Unicast is active
Gateway Address Family: IPv6 Unicast
Table ID: 0xe0800012 Gateway Reference Count: 4 Gateway AF Bits : 0x1 Nexthop Count: 2
Critical Trigger Delay: 50msec
Non-critical Trigger Delay: 10000msec

Nexthop Version: 3, RIB version: 1
EPE Table Version: 1, EPE Label version: 0
EPE Downloaded Version: 0, EPE Standby Version: 0

Status codes: R/UR Reachable/Unreachable
C/NC Connected/Not-connected L/NL Local/Non-local


Next Hop PR I Pending Registration Invalid (Policy drop)
Status Metric

Tbl-ID

Notf

LastRIBEvent

RefCount
2001:DB8:2::1  [R][C][NL] 0 e0800012 1/0 00:37:03 (Cri) 1/3
2001:DB9:2::3  [R][C][NL] 0 e0800012 1/0 00:37:03 (Cri) 1/4
Counters
Next Hop Reachable Unreachable MetricIncrease MetricDecrease
2001:DB8:2::1 1 0 0 0
2001:DB9:2::3 1 0 0 0
```

Step 6 Verify IPv6 nexthop for the eBGP route.

Example:

```
Router# show bgp vrf RFC5549 ipv4 unicast nexthops 2001:DB8:2::1 Nexthop: 2001:DB8:2::1
VRF: RFC5549
Nexthop ID: 0x6000001, Version: 2 Nexthop Flags: 0x00020082 Nexthop Handle: 0x211b390
Tree Nexthop Handle: 0x211b390

Advertising neighbors:
2001:DB8:2::1

RIB Related Information:
Firsthop interface handle 0x008001e0
Gateway TBL Id: 0xe0800012 Gateway Flags: 0x00000080 Gateway Handle: 0x24c35a8
Gateway: reachable, Connected route, prefix length 64 Resolving Route: 2001:DB8:2::/64 (connected)
Paths: 1
RIB Nexhop ID: 0x20010 Nexthop sync slot: 23
Status: [Reachable][Connected][Not Local] Metric: 0
ORR afi bits: 0x0
Registration: Asynchronous, Completed: 00:37:16 Events: Critical (1)/Non-critical (0)
```

```
Last Received: 00:37:16 (Critical)
Last gw update: (Crit-notif) 00:37:16(rib)
Reference Count: 1

Reachable Notifications: 1 (last at Mar 27 23:06:16.748)
Unreachable Notifications: 0
Metric Increase Notifications: 0
Metric Decrease Notifications:
Most Recent Events: 0
Time Event Type Metric
Mar 27 23:06:16.748 Reachable 0

Prefix Related Information Active Tables: [IPv4 Unicast] Metrices: [0xffffffff] Reference Counts:
[1] Encapsulations: []
Interface Handle: 0x0

Linked Nexthop Count: 1 Attr ref-count: 3
```

Step 7 Verify IPv4 nexthop for the eBGP route.

Example:

```
Router# show bgp vrf RFC5549 ipv4 unicast nexthops 2001:DB9:2::3
Nexthop: 2001:DB9:2::3
VRF: RFC5549
Nexthop ID: 0x6000002, Version: 3 Nexthop Flags: 0x00000082 Nexthop Handle: 0x211b130
Tree Nexthop Handle: 0x211b130

RIB Related Information:
Firsthop interface handle 0x00800240
Gateway TBL Id: 0xe0800012 Gateway Flags: 0x00000080 Gateway Handle: 0x24c34f0
Gateway: reachable, Connected route, prefix length 64 Resolving Route: 2001:DB9:2::/64 (connected)
Paths: 0
RIB Nexhop ID: 0x20011 Nexthop sync slot: 30
Status: [Reachable][Connected][Not Local] Metric: 0
ORR afi bits: 0x0
Registration: Asynchronous, Completed: 00:37:24 Events: Critical (1)/Non-critical (0)
Last Received: 00:37:24 (Critical)
Last gw update: (Crit-notif) 00:37:24(rib)
Reference Count: 1

Reachable Notifications: 1 (last at Mar 27 23:06:16.748)
Unreachable Notifications: 0
Metric Increase Notifications: 0
Metric Decrease Notifications:
Most Recent Events: 0
Time Event Type Metric
Mar 27 23:06:16.748 Reachable 0

Prefix Related Information Active Tables: [IPv4 Unicast] Metrices: [0xffffffff] Reference Counts:
[1] Encapsulations: []
Interface Handle: 0x0 Attr ref-count: 4
```

Step 8 Verify eBGP and iBGP routes in RIB.

a) Display routing information for the specified prefix in the selected VRF.

Example:

```
Router# show route vrf RFC5549 192.1.0.1/32
Routing entry for 192.1.0.1/32
Known via "bgp 2", distance 20, metric 0
Tag 1
Local Label 24004, type external Installed Mar 27 23:37:00.244 for 00:09:00 Routing Descriptor
Blocks
```

```
fe80::83:9eff:fe38:c59f, from 2001:DB8:2::1, via GigabitEthernet0/1/0/0.2, BGP external Nexthop
in Vrf: "RFC5549", Table: "default", IPv6 Unicast, Table Id: 0xe0800012 Route metric is 0
No advertising protos.


Router# show route vrf RFC5549 192.1.0.3/32
Routing entry for 192.1.0.3/32
Known via "bgp 2", distance 200, metric 0
Local Label 24007, type internal Installed Mar 27 23:36:47.287 for 00:09:28 Routing Descriptor
Blocks
2001:DB9:2::3, from 2001:DB9:2::3
Nexthop in Vrf: "RFC5549", Table: "default", IPv6 Unicast, Table Id: 0xe0800012 Route metric is
0
No advertising protos.
```

b) Display detailed routing information for the specified prefix in the selected VRF.

Example:

```
Router# show route vrf RFC5549 192.1.0.3/32 detail
Routing entry for 192.1.0.3/32
Known via "bgp 2", distance 20, metric 0
Tag 1, type external
Installed Mar 27 23:37:00.244 for 00:09:32 Routing Descriptor Blocks
fe80::83:9eff:fe38:c59f, from 2001:DB8:2::1, via GigabitEthernet0/1/0/0.2, BGP external Nexthop
in Vrf: "RFC5549", Table: "default", IPv6 Unicast, Table Id: 0xe0800012 Route metric is 0
Label: None Tunnel ID: None
Binding Label: None
Extended communities count: 0 NHID: 0x0 (Ref: 0)
Path Grouping ID: 1 Route version is 0x2 (2) Local Label: 0x5dc4 (24004) IP Precedence: Not Set
QoS Group ID: Not Set Flow-tag: Not Set Fwd-class: Not Set
Route Priority: RIB_PRIORITY_RECURSIVE (10) SVD Type RIB_SVD_TYPE_LOCAL
Download Priority 3, Download Version 59 No advertising protos.


Router# show route vrf RFC5549 192.1.0.3/32 detail
Routing entry for 192.1.0.3/32
Known via "bgp 2", distance 200, metric 0, type internal Installed Mar 27 23:36:47.287 for 00:09:52
Routing Descriptor Blocks
23:2:2::3, from 23:2:2::3
Nexthop in Vrf: "RFC5549", Table: "default", IPv6 Unicast, Table Id: 0xe0800012 Route metric is
0
Label: None Tunnel ID: None
Binding Label: None
Extended communities count: 0 NHID: 0x0 (Ref: 0)
Path Grouping ID: 2 Route version is 0x2 (2) Local Label: 0x5dc7 (24007) IP Precedence: Not Set
QoS Group ID: Not Set Flow-tag: Not Set Fwd-class: Not Set
Route Priority: RIB_PRIORITY_RECURSIVE (10) SVD Type RIB_SVD_TYPE_LOCAL
Download Priority 3, Download Version 57 No advertising protos.
```

Step 9 Verify eBGP and iBGP routes in Cisco Express Forwarding (CEF).

a) Display CEF information for the specified prefix in the selected VRF.

Example:

```
Router# show cef vrf RFC5549 192.1.0.1/32
192.1.0.1/32, version 59, internal 0x1000001 0x30 (ptr 0x61c125a8) [1], 0x600 (0x6379bc58), 0xa20
 (0x632aa3b8)
Updated Mar 27 23:37:00.247
remote adjacency to GigabitEthernet0/1/0/0.2
Prefix Len 32, traffic index 0, precedence n/a, priority 3
gateway array (0x6441c190) reference count 2, flags 0x8068, source rib (7), 0 backups [3 type 4
flags 0x108401 (0x632c83b0) ext 0x0 (0x0)]
LW-LDI[type=1, refc=1, ptr=0x6379bc58, sh-ldi=0x632c83b0] gateway array update type-time 1 Mar 27
 23:37:00.247
```

```
LDI Update time Mar 27 23:37:00.247
LW-LDI-TS Mar 27 23:37:00.247
via fe80::83:9eff:fe38:c59f/128, GigabitEthernet0/1/0/0.2, 4 dependencies, weight 0, class 0,
bgp-ext [flags 0x6020]
path-idx 0 NHID 0x0 [0x632e75a0 0x0]
next hop VRF - 'RFC5549', table - 0xe0800012 next hop fe80::83:9eff:fe38:c59f/128
remote adjacency
local label 24004 labels imposed {None} Load distribution: 0 (refcount 3)
Hash OK Interface Address
0 Y GigabitEthernet0/1/0/0.2 remote

Router# show cef vrf RFC5549 192.1.0.3/32
192.1.0.3/32, version 57, internal 0x1000001 0x30 (ptr 0x61c12698) [1], 0x0 (0x0), 0x200
(0x632aa1d8)
Updated Mar 27 23:36:47.289
remote adjacency to GigabitEthernet0/1/0/1.2
Prefix Len 32, traffic index 0, precedence n/a, priority 3
gateway array (0x6441c098) reference count 2, flags 0xa078, source rib (7), 0 backups [1 type 4
flags 0x148501 (0x632c8358) ext 0x0 (0x0)]
LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
gateway array update type-time 1 Mar 27 23:36:47.289
LDI Update time Mar 27 23:36:47.367
via 2001:DB9:2::3/128, 2 dependencies, recursive [flags 0x6000]
path-idx 0 NHID 0x0 [0x61c7a398 0x0]
next hop VRF - 'RFC5549', table - 0xe0800012 next hop 2001:DB9:2::3/128 via 2001:DB9:2::3/128
local label 24007
next hop 2001:DB9:2::3/128 Gi0/1/0/1.2 labels imposed {None}

Load distribution: 0 (refcount 1)
Hash OK Interface Address
0 Y GigabitEthernet0/1/0/1.2 remote
```

b) Display detailed CEF information for the specified prefix in the selected VRF.

Example:

```
Router# show cef vrf RFC5549 192.1.0.1/32 detail
192.1.0.1/32, version 59, internal 0x1000001 0x30 (ptr 0x61c125a8) [1], 0x600 (0x6379bc58), 0xa20
 (0x632aa3b8)
Updated Mar 27 23:37:00.247
remote adjacency to GigabitEthernet0/1/0/0.2
Prefix Len 32, traffic index 0, precedence n/a, priority 3
gateway array (0x6441c190) reference count 2, flags 0x8068, source rib (7), 0 backups [3 type 4
flags 0x108401 (0x632c83b0) ext 0x0 (0x0)]
LW-LDI[type=1, refc=1, ptr=0x6379bc58, sh-ldi=0x632c83b0] gateway array update type-time 1 Mar 27
 23:37:00.247
LDI Update time Mar 27 23:37:00.247
LW-LDI-TS Mar 27 23:37:00.247
via fe80::83:9eff:fe38:c59f/128, GigabitEthernet0/1/0/0.2, 4 dependencies, weight 0, class 0,
bgp-ext [flags 0x6020]
path-idx 0 NHID 0x0 [0x632e75a0 0x0]
next hop VRF - 'RFC5549', table - 0xe0800012 next hop fe80::83:9eff:fe38:c59f/128
remote adjacency
local label 24004 labels imposed {None}

Load distribution: 0 (refcount 3)

Hash OK Interface Address
0 Y GigabitEthernet0/1/0/0.2 remote

Router# show cef vrf RFC5549 192.1.0.3/32 detail
192.1.0.3/32, version 57, internal 0x1000001 0x30 (ptr 0x61c12698) [1], 0x0 (0x0), 0x200
(0x632aa1d8)
Updated Mar 27 23:36:47.289
remote adjacency to GigabitEthernet0/1/0/1.2
```

```
Prefix Len 32, traffic index 0, precedence n/a, priority 3
gateway array (0x6441c098) reference count 2, flags 0xa078, source rib (7), 0 backups [1 type 4
flags 0x148501 (0x632c8358) ext 0x0 (0x0)]
LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
gateway array update type-time 1 Mar 27 23:36:47.289
LDI Update time Mar 27 23:36:47.367
via 2001:DB9:2::3/128, 2 dependencies, recursive [flags 0x6000] path-idx 0 NHID 0x0 [0x61c7a398
0x0]
next hop VRF - 'RFC5549', table - 0xe0800012 next hop 2001:DB9:2::3/128 via 2001:DB9:2::3/128
local label 24007
next hop 2001:DB9:2::3/128 Gi0/1/0/1.2 labels imposed {None}

Load distribution: 0 (refcount 1)

Hash OK Interface Address
0 Y GigabitEthernet0/1/0/1.2 remote
```

BGP MDT address family sessions

A multicast distribution tree (MDT) address family session in BGP is a feature that

- enables you to exchange IPv4 multicast routing information using the MDT subaddress family identifier (SAFI)
- supports multicast virtual private networks (MVPN), including both IPv4 and IPv6 deployments, and
- allows you to redistribute multicast routing information across multiple protocols and VRFs.

This feature is essential for service providers and enterprises that deploy multicast VPNs over BGP-based core networks.

Configure a BGP MDT address family session

Set up a BGP session for IPv4 MDT, which can also support multicast VPNv4 and VPNv6 distribution.

Before you begin

- Ensure you have decided on the autonomous system number (ASN) and router ID.
- Identify the interface or loopback address to use as the update source.

Procedure

Step 1 Enter BGP configuration mode for your autonomous system.

Example:

```
Router# configure
Router(config)# router bgp 120
```

Step 2 Enable the MDT address family. If you are configuring multicast MVPN, also enable the unicast and VPN address families.

Example:

```
Router(config-vrf)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# address-family vpnv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# address-family ipv4 mdt
Router(config-bgp-af)# exit
```

Step 3 Configure the BGP neighbor, specify its remote AS, and set the update source.

Example:

```
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# update-source loopback 0
```

Step 4 Activate the required address families for the neighbor.

Example:

```
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# address-family vpnv4 unicast
Router(config-bgp-nbr-af)# exit
Router(config-bgp)# address-family ipv4 mdt
Router(config-bgp-af)# exit
```

Step 5 If you are configuring multicast MVPN, create a VRF instance and configure a route distinguisher (RD).

Example:

```
Router(config-bgp)# vrf vpn1
Router(config-bgp-vrf)# rd 1:1
```

Step 6 Within the VRF, redistribute routes from another routing protocol, such as EIGRP or OSPF, into BGP.

Example:

```
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# redistribute eigrp 23
```

Step 7 Commit the configuration changes.

Example:

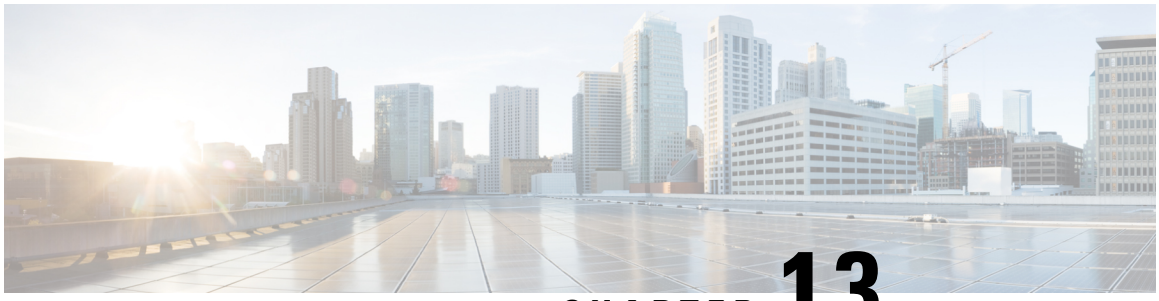
```
Router(config-bgp-vrf-af)#commit
```

Step 8 Run the **show running-config** command to verify the configuration.

Example:

```
Router# show running-config
router bgp 120
 address-family ipv4 mdt
 !
 address-family vpnv4 unicast
 !
 vrf vpn1
  rd 1:1
  address-family ipv4 unicast
   redistribute eigrp 23
  !
 !
```

```
neighbor 172.168.40.24
 remote-as 2002
 update-source Loopback0
 address-family ipv4 mdt
 !
 address-family vpnv4 unicast
 !
 address-family ipv4 unicast
 !
!
```



CHAPTER 13

BGP Dynamic Neighbors and Resource Management

This chapter explains how to configure, manage, and monitor BGP dynamic neighbors and resource allocation. It covers advanced neighbor management, such as managing dynamic peering, configuring multi-instance BGP, resetting neighbors, and managing resource allocation.

BGP dynamic neighbors

A BGP dynamic neighbor is a peer that is discovered and established using a configured IP address range, instead of an explicitly configured or static neighbor configuration.

Dynamic neighbor configuration allows a router to peer with multiple remote neighbors without explicit static entries for each one. The router accepts BGP connections from any peer whose IP address falls within a specified subnet range.

Example:

In a data center, you can configure one dynamic neighbor range to handle a pool of servers or routers that may change frequently.

Benefits of BGP dynamic neighbor support

BGP dynamic neighbor support provides these benefits:

- Allows you to configure each range of IP addresses as a subnet.
- Reduces CLI configuration complexity in large networks.
- Supports both IPv4 and IPv6 peering.

Configure BGP dynamic neighbors using address range

Purpose: Configure BGP dynamic neighbors using an IP prefix, reducing manual neighbor statements in large-scale networks.

Context: Use this when you want to accept BGP peerings from a set of IP addresses, such as a subnet, instead of individually configuring each neighbor.

This figure illustrates a sample topology.

Figure 10: Dynamic neighbor connection topology

After you configure the subnet range for dynamic neighbors on Router A, Router B can initiate a TCP session from any IP address within that range. Router A then automatically establishes the BGP neighbor connection with Router B, with no further CLI configuration required after the initial setup.

In this example, local autonomous system (AS) is 100 and remote AS is 1.

Follow these steps to configure BGP dynamic neighbors using an address range.

Before you begin

- Identify the IP subnet that should be allowed as dynamic neighbors.
- Ensure you know the remote AS number and the local interface for source IP.

Procedure

Step 1 Enter BGP configuration mode with your AS number and configure the dynamic neighbor prefix.

Example:

```
Router#configure
Router(config)# router bgp 100
Router(config-bgp)# neighbor 10.0.0.0/16
```

Step 2 Assign the remote AS number and specify the update source interface.

Example:

```
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# update-source FourHundredGige 0/0/0/0
```

Step 3 Enable the address family for IPv4 unicast and commit the configuration.

Example:

```
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr)# commit
```

Step 4 Verify the active configuration.

Example:

```
Router# show running-config
router bgp 100
 neighbor 10.0.0.0/16
  remote-as 1
  update-source FourHundredGige 0/0/0/0
  address-family ipv4 unicast
 !
!
```

The router accepts BGP sessions from any peer with an IP address within the specified range, creating dynamic neighbor entries automatically.

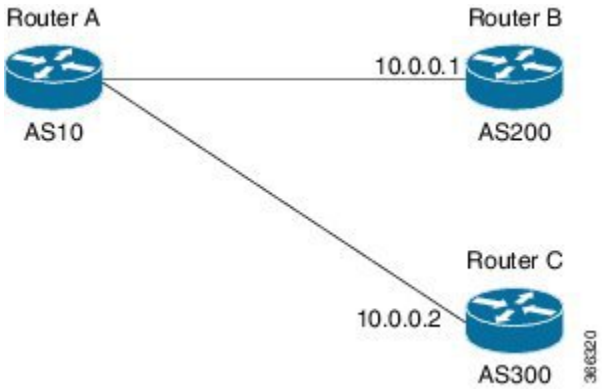
Configure remote AS list

Configure a list of remote autonomous system (AS) numbers for dynamic BGP neighbors.

Context: Use this procedure when you have dynamic neighbors from multiple remote ASes and want to control which AS numbers are accepted.

This figure illustrates a sample topology.

Figure 11: Connection topology with dynamic neighbors from multiple remote ASes



In this example topology, Router B and Router C are configured as remote BGP peers. Both Router B and Router C are in different autonomous systems.

A list of the autonomous systems of the remote routers is created and the list is then configured in Router A under neighbor mode using the **remote-as-list** command.

Before you begin

- Identify the IP subnet that should be allowed as dynamic neighbors.
- Ensure you know the remote AS number and the local interface for source IP.

Procedure

Step 1 Enter BGP configuration mode and define your local AS number.

Example:

```
Router#configure
Router(config)# router bgp 100
Router(config-bgp)#
```

Step 2 Run the **as-list** command to create an AS list.

Example:

```
Router(config-bgp)#as-list test
Router(config-bgp-as-list)#200
Router(config-bgp-as-list)#300
Router(config-bgp-as-list)#exit
Router(config-bgp)#
```

Step 3 Run the **neighbor address/prefix** command to configure the dynamic neighbor prefix.

Example:

```
Router(config-bgp)# neighbor 10.0.0.0/16
```

Step 4 Run the **remote-as-list** command to assign the remote AS list to the neighbor.

Example:

```
Router(config-bgp-nbr)# remote-as-list test
```

Step 5 Enable the address family for IPv4 unicast and commit the configuration.

Example:

```
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr)# commit
```

Dynamic neighbor sessions are accepted only if their AS number is in the configured remote AS list.

Configure maximum peers and idle-watch timeout

Limit the number of dynamic BGP neighbors and set the idle timeout for TCP instances.

Context: Use these settings to manage router resources and avoid excessive neighbor creation.

Before you begin

None

Procedure

Step 1 Enter BGP configuration mode with your AS number and configure the dynamic neighbor prefix.

Example:

```
Router#configure
Router(config)# router bgp 100
Router(config-bgp)# neighbor 10.0.0.0/16
```

Step 2 Run the **maximum-peers** command to set the maximum number of dynamic neighbors for this neighbor address range.

Example:

```
Router(config-bgp-nbr)# maximum-peers 16
```

Step 3 Run the **idle-watch-time** command to set the idle-watch timeout in seconds and save the configuration.

Example:

```
Router(config-bgp-nbr)# idle-watch-time 120
Router(config-bgp-nbr)# commit
```

The router limits dynamic neighbor creation for the range and removes idle TCP sessions after the specified time.

Multi-instance and multi-AS BGP

Multi-instance and multi-AS BGP is a configuration approach that

- allows each BGP instance to use a unique AS number
- provides address family isolation by mapping different address families to separate BGP instances, and
- enables higher scaling and more granular resource management by distributing peer sessions and BGP tables across instances.

Restrictions and guidelines for multi-instance BGP

This section outlines the restrictions and provides guidelines for configuring and managing multi-instance BGP on a router.

Instance limits and identification

- A router supports a maximum of four BGP instances.
- Each BGP instance must have a unique router ID.

Address family configuration

- Only one address family can be configured under each BGP instance, except VPNv4, VPNv6, and RT-Constrain.
- IPv4/IPv6 unicast and their labeled variants must reside in the same BGP instance.
- IPv4/IPv6 multicast must reside in the same BGP instance as their unicast counterparts.

Configuration management

All configuration changes for a single instance can be committed together; however, changes for multiple instances must be committed separately.

Guidelines

We recommend using unique update-source interfaces in the default VRF over all instances when peering with the same remote router.

BGP neighbor resets

BGP neighbor reset is an administrative process that

- refreshes or clears BGP sessions to apply configuration changes
- can be performed as a soft or hard reset, and
- helps maintain accurate routing information by forcing policy re-evaluation or session re-establishment.

These resets are primarily categorized into two types, each with a distinct impact on the BGP session:

- Soft reset: Resets only routing information without tearing down the TCP session.
- Hard reset: Tears down the TCP session and removes all routes learned from the neighbor.

Reset BGP neighbors using inbound soft reset

Refresh inbound BGP policies and routes without resetting the entire session.

Context: Use this when inbound policy or attribute changes need to be applied to BGP-learned routes.

Before you begin

None

Procedure

Step 1 Run the **show bgp neighbors** command to verify the route refresh capability from the neighbor is enabled.

Example:

```
Router# show bgp neighbors
```

Step 2 Run the **clear bgp** command to initiate the inbound soft reset.

Example:

```
Router# clear bgp ipv4 unicast 10.0.0.1 soft in
```

- Use * instead of specific IP address to reset all neighbors.
- Use an AS number or *external* keyword for group resets.

The router requests the neighbor to resend routes, applying updated inbound policies.

Reset BGP neighbors using outbound soft reset

Resend all outbound BGP routes and policies to a neighbor.

Context: Use when outbound policy or attributes have changed.

Before you begin

None

Procedure

Step 1 Run the **show bgp neighbors** command to verify the route refresh capability from the neighbor is enabled.

Example:

```
Router# show bgp neighbors
```

Step 2 Run the **clear bgp** command to initiate the outbound soft reset.

Example:

```
Router# clear bgp ipv4 unicast 10.0.0.1 soft out
```

- Use * instead of specific IP address to reset all neighbors.
- Use an AS number or *external* keyword for group resets.

The router resends all routes for the specified address family to the neighbor.

Reset BGP neighbors using hard reset

Completely resets the TCP connection and clears all routing information from a BGP neighbor.

Context: Use if you need to force a full session re-establishment or resolve persistent connectivity issues.

A hard reset removes the TCP connection to the neighbor, removes all routes received from the neighbor from the BGP table, and then re-establishes the session with the neighbor. If the *graceful* keyword is specified, the routes from the neighbor are not removed from the BGP table immediately, but are marked as stale. After the session is re-established, any stale route that has not been received again from the neighbor is removed.

Before you begin

None

Procedure

Run the **clear bgp** command with optional *graceful* and *soft* keywords.

Example:

```
Router# clear bgp ipv4 unicast 10.0.0.3 graceful soft out
```

- *graceful*: The *graceful* keyword preserves routes temporarily until the neighbor re-establishes.
- Use * instead of specific IP address to reset all neighbors.
- Use an AS number or *external* keyword for group resets.

The TCP session is terminated and re-established, and routing tables are refreshed.

Clear BGP caches, tables, and databases

Remove all entries from specified BGP data structures and reset neighbor sessions.

Context: Use for troubleshooting or when data integrity is in question.

Perform this procdure to remove all contents of a particular cache, table, or database. This can become necessary when the contents of the particular structure have become, or are suspected to be, invalid.

Before you begin

Try Reset BGP neighbors using hard reset to resolve the issue.

Procedure

Run the **clear bgp** command to clear BGP caches and sessions.

Example:

```
Router# clear bgp ipv4 172.20.1.1
```

- **clear bgp ipv4** <*ip address*> : This command clears the caches, tables, and databases for a specific neighbor.
- **clear bgp** *external*: This command clears the caches, tables, and databases for all external peers.
- **clear bgp** *: This command clears the caches, tables, and databases for all BGP neighbors.

The selected sessions, caches, and databases are reset or cleared.

Disable a BGP neighbor

Administratively shut down a BGP neighbor session without removing its configuration.

Context: Use this procedure to temporarily disable a peer, for maintenance or troubleshooting.

Before you begin

None

Procedure

Step 1 Run the **configure** command to enter global configuration mode.

Example:

```
Router#configure
```

Step 2 Enter BGP configuration mode with your AS number and specify the neighbor.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 172.16.40.24
```

Step 3 Run the **shutdown** command to shut down the neighbor and save the configuration.

Example:

```
Router(config-bgp-nbr)# shutdown
Router(config-bgp-nbr)# commit
```

Disables all active sessions for the specified neighbor, leaving the session disabled and still in the configuration.

Neighbor capability suppressions

Neighbor capability suppression is a BGP feature that

- disables capability negotiation between BGP peers during the Open message exchange
- ensures interoperability with legacy devices that do not support the Capabilities option, and
- allows BGP sessions to be established even if the peer does not understand capabilities negotiation.

In BGP, capability negotiation is a mechanism that allows peers to exchange information about supported protocol extensions during session establishment. By default, BGP peers use this feature to agree on features they both support. However, some older customer premises equipment (CPE) devices do not recognize the Capabilities option, which can prevent successful BGP session setup. The neighbor capability suppression feature disables this negotiation in the Open message, enabling compatibility with such legacy devices.

Suppress BGP neighbor capabilities

Disable BGP capability negotiation during the Open message exchange for compatibility with legacy devices.

Context: Use when peering with devices that do not support BGP capabilities negotiation.

Before you begin

None

Procedure

Step 1 Run the **configure** command to enter global configuration mode.

Example:

```
Router#configure
```

Step 2 Enter BGP configuration mode with your AS number and specify the neighbor.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 172.16.40.24
```

Step 3 Run the **capability** *suppress all* command to suppress all capabilities to the neighbor and save the configuration.

Example:

```
Router(config-bgp-nbr)# capability suppress all
Router(config-bgp-nbr)# commit
```

Capabilities negotiation is disabled for the specified neighbor.

Display BGP system and network statistics

View BGP routing tables, neighbor information, and performance statistics for troubleshooting and network management.

Context: Use these commands to monitor BGP operation and verify configuration status.

You can use any of these commands to display specific BGP information as needed:

Before you begin

None

Procedure

Step 1 Run the **show bgp cidr-only** command to display routes with non-natural network masks or Classless Inter-domain Routing (CIDR).

Example:

```
Router# show bgp cidr-only
```

Step 2 Run the **show bgp community** command to display routes that match a specific BGP community.

Example:

```
Router# show bgp community 1081:5 exact-match
```

Step 3 Run the **show bgp regexp regular-expression** command to display routes matching an AS path regular expression.

Example:

```
Router# show bgp regexp "^3 "
```

Step 4 Run the **show bgp** command to display all entries in the BGP routing table.

Example:

```
Router# show bgp
```

Step 5 Run the **show bgp neighbors** *ip-address [advertised-routes | dampened-routes | flap-statistics | performance-statistics | received prefix-filter | routes]* command to display neighbor information.

The following keywords display specific BGP neighbor information:

- *advertised-routes*: all routes that the router advertised to the neighbor
- *dampened-routes*: dampened routes learned from the neighbor
- *flap-statistics*: flap statistics for routes learned from the neighbor
- *performance-statistics*: performance statistics for the BGP process and this neighbor
- *received prefix-filter*: the received prefix list filter
- *routes*: routes learned from the neighbor

Example:

```
Router# show bgp neighbors 10.0.101.1
```

Step 6 Run the **show bgp paths** command to display all BGP paths in the database.

Example:

```
Router# show bgp paths
```

Step 7 Run the **show bgp neighbor-group** *group-name* **configuration** command to display the effective configuration for a neighbor group, including any configuration inherited by this neighbor group.

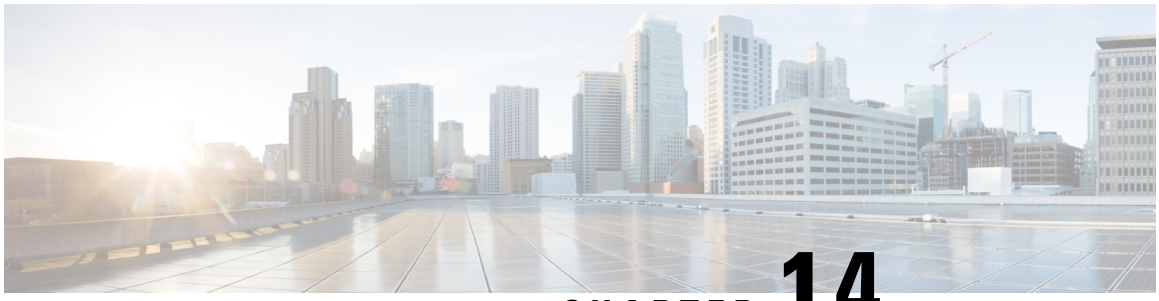
Example:

```
Router# show bgp neighbor-group group_1 configuration
```

Step 8 Run the **show bgp summary** command to display the summary status of all BGP connections.

Example:

```
Router# show bgp summary
```



CHAPTER 14

BGP Multipath and Load Balancing Techniques

This chapter describes the features that enable BGP multipath and load-balancing. These techniques allow you to optimize performance and resilience in complex, large-scale environments.

BGP multipath and load balancing techniques are a set of network solutions that

- enable traffic to use multiple equal-cost paths for improved reliability,
- increase scalability and efficiency by distributing traffic across iBGP and eBGP sessions, and
- support advanced policies and resilient hashing for granular control.

EIBGP policy-based multipath with equal-cost multipath

EIBGP policy-based multipath with Equal-Cost Multipath (ECMP) is a BGP feature that

- enables routers to use multiple best paths for forwarding traffic,
- allows load balancing across both internal (iBGP) and external (eBGP) BGP sessions, and
- supports advanced policies like policy-based multipath, ECMP, resilient hashing, and selective multipath.

By using BGP communities, nexthops, path types, and consistent hashing, the solution increases flexibility and reliability in managing traffic distribution across large-scale networks

- ECMP: A routing strategy that allows load balancing over multiple paths with the same cost.
- eBGP/iBGP/eiBGP: External/Internal/External-Internal BGP sessions, defining the relationship between the BGP peers.
- BGP community: An attribute used to tag and group routes for policy application.

Table 19: Feature History Table

Feature Name	Release Name	Description
EIBGP policy-based multipath with equal-cost multipath	Release 7.10.1	You can control traffic distribution and load balancing in BGP by enabling policy-based multipath selection for iBGP, eBGP, and eiBGP sessions. This approach uses BGP communities, nexthops, and path types to define how paths are selected. Additionally, by using the Equal-Cost Multipath (ECMP) option in eiBGP, you can balance traffic across multiple eligible iBGP paths chosen for eiBGP. This feature gives you greater flexibility and efficiency in managing BGP routing and load balancing. The feature introduces these changes: CLI: The keywords **route-policy** and **equal-cost** are added to the **maximum-paths** command. YANG Data Model: `Cisco-IOS-XR-um-router-bgp-cfg` (see GitHub, YANG Data Models Navigator)

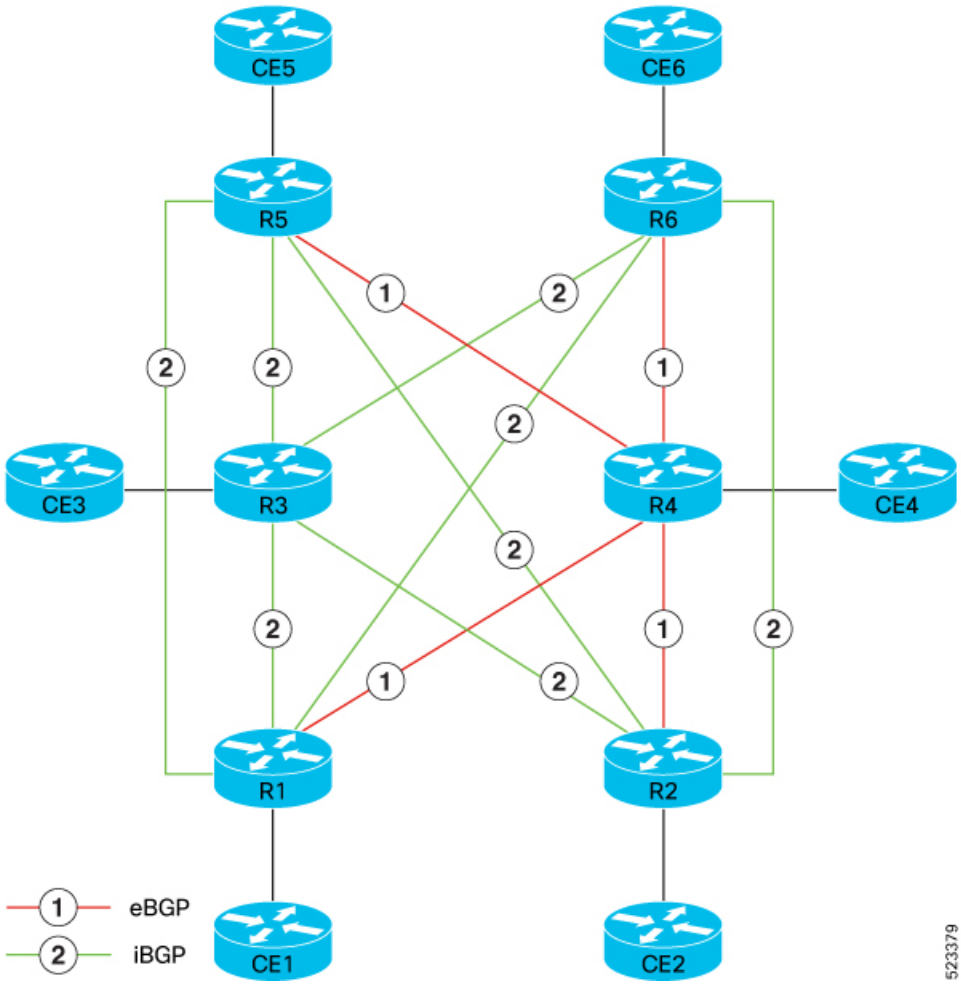
The enhanced policy-based multipath selection in BGP operates now at the default Virtual Routing and Forwarding (VRF) level for variations of BGP, such as iBGP, eBGP, and eiBGP. To improve this functionality, the policy-based multipath selection is now extended to include iBGP, eBGP, and eiBGP by utilizing communities as the underlying mechanism. By utilizing communities, the selection of multiple paths based on specific policy criteria becomes more elaborate. It enables better control over the routing decisions within the BGP network.

eiBGP traditionally implements the unequal-cost mutipath (UCMP) capability to enable the use of both iBGP and eBGP paths. This feature, utilizing the equal-cost multipath option (ECMP), ensures that the nexthop IGP metric remains consistent across the chosen iBGP paths. Hence the metric evaluation is not performed between eBGP and iBGP paths because they have distinct path types.

Example topology and behavior

This topology illustrates a network comprising BGP peers denoted as R1 through R6.

Figure 12: Example topology



Consider a scenario where there is a specific need to transition from using eBGP multipaths to iBGP multipaths. Throughout this transition, you require the simultaneous operation of both eBGP and iBGP to facilitate a seamless migration.

Topology setup

This topology showcases distinct path types, where eBGP paths are visually depicted using a red-colored line labeled as 1, and the iBGP paths are visually illustrated using a green-colored line labeled as 2.

Classic eiBGP behavior

In the context of CE routers, CEI to CE6, the preferred path for prefixes will be from eBGP, specifically from the R4 router. The selection of best paths prioritizes eBGP multipaths from R4, although paths might exist from R5 and R6 routers and also from RI and R2 routers through iBGP. This is the classic behavior. In classic eiBGP, unequal cost paths are employed, leading to the disregard of metrics. However, you rely on the IGP metric for optimal performance.

Change in eiBGP behavior

The iBGP paths with the shortest AS-PATH length are chosen for R5 and R6. The same iBGP multipath selection process applies to paths from R1 and R2. As a result, R1 and R2 establish iBGP peering sessions

with R3. This setup creates a combination of eBGP and iBGP paths, called eiBGP, which are available for prefixes advertised to hosts beyond the CE devices. The CE routers must balance prefixes between R3 and R4, and you should exclude paths from R5, R6, R1, and R2. To do this, configure the additive community attribute on R1 and R2 for routes advertised toward R5 and R6.

With this topology, you can run both eBGP and iBGP, allowing a smooth transition between eBGP and iBGP multipaths. Include the default VRF in policy-based multipath selection to apply route policies that control how your network distributes traffic. Use BGP attributes such as communities, next hops, and path types in these policies to select paths. For example, use BGP communities to prioritize routes or change next hops to direct traffic over specific paths. This approach helps you optimize routing decisions, control traffic distribution, and improve load balancing across all BGP types in your network.

Enable ECMP to distribute traffic evenly across multiple equal-cost paths. This prevents overloading any single path and improves load balancing. With the ECMP option in eiBGP, the router can use multiple iBGP paths with equal cost for traffic distribution.

Benefits of EIBGP policy-based multipath with ECMP

- Optimizes traffic distribution and network utilization across iBGP, eBGP, and eiBGP sessions.
- Enables seamless migration between eBGP and iBGP multipath scenarios.
- Provides granular control over path selection using communities, nexthops, and path types.
- Ensures efficient load balancing and prevents single-path overload by leveraging ECMP.

Caveats of multipath selection without BGP attributes and ECMP

- If BGP communities, nexthops, or path types are not used in policy-based multipath selection, control over routing is reduced, leading to suboptimal load balancing.
- Not enabling ECMP in eiBGP causes the router to rely on classic best-path selection and forgo the benefits of multipath load balancing.
- Avoid comparing IGP metrics between eBGP and iBGP paths, as ECMP only considers iBGP paths with equal cost.

Restrictions and guidelines for eiBGP policy-based multipath with ECMP

eiBGP multipath configuration restrictions

- You cannot configure eiBGP along with either eBGP or iBGP.
- The maximum-paths route policy checks only the community, next hop, and path type.
- The OpenConfig model is not supported.

eiBGP multipath configuration guidelines

- Use the Accumulated Interior Gateway Protocol (AIGP) metric attribute only with equal-cost eiBGP paths.
- When you configure both eBGP and iBGP multipath, you can assign the same or different route policies to each. The router automatically applies the policy for the best path type of each prefix:

- If iBGP provides the best path for a prefix, the iBGP route policy is applied.
- If eBGP provides the best path, the eBGP route policy is applied.

How eiBGP policy-based multipath with ECMP works

Summary

The key components involved in the process are:

- BGP communities: Used for tagging and grouping routes according to policy.
- Multipath selection policy: Determines which iBGP, eBGP, or eiBGP paths are included based on community, nexthop, and path type.
- ECMP option: Ensures that equal-cost iBGP paths are included in eiBGP multipath selection.

Enhanced policy-based multipath selection operates at the default VRF for iBGP, eBGP, and eiBGP. The solution uses BGP communities to enable flexible and granular selection of multiple paths according to policy.

Workflow

These stages describe how eiBGP policy-based multipath with ECMP works:

1. The router receives routes from eBGP and iBGP peers.
2. Route policies using communities are applied to select eligible multipaths.
3. With ECMP enabled, the router balances traffic across all equal-cost iBGP paths that are chosen as part of the eiBGP multipath set.
4. The router installs the selected multipaths and forwards traffic accordingly.

Result

Traffic distribution and load balancing are controlled and optimized, supporting seamless transitions between eBGP and iBGP multipath operation.

Configure eiBGP policy-based multipath with ECMP

Enable eiBGP policy-based multipath routing with ECMP by applying route policies that use communities, path types, or next hops.

Use this task to configure ECMP for eiBGP, allowing your routers to consider multiple eligible paths for load balancing, based on policy.

Before you begin

- Ensure that BGP and required address families are enabled on all participating routers.
- Have appropriate community values and route policies planned.

Follow these steps to configure eiBGP policy-based multipath with ECMP.

Procedure

Step 1 Define a community set for R1 and R2 routers.

Example:

```
Router(config)# community-set ABC
Router(config-comm)# 2:1
Router(config-comm)# end-set
```

Step 2 Create a route policy named EIBGP on R1 and R2.

Example:

```
Router(config)# route-policy EIBGP
Router(config-rpl)# if community matches-any ABC 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
```

This policy checks the BGP communities of received routes and takes action based on the community value.

- If the community matches "ABC", the route is eligible for multipath.
- If not, the route is dropped from multipath consideration.
- You can also match on path-type or next-hop in the route policy as needed.

Step 3 Configure BGP to use ECMP for eiBGP and apply the route policy.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# maximum-paths eibgp 32 equal-cost route-policy EIBGP
Router(config-bgp-af)# commit
```

Step 4 Verify the running configuration.

Example:

```
Router# show running-config
community-set ABC
 2:1
 end-set
!

route-policy EIBGP
  if community matches-any ABC then
    pass
  else
    drop
  endif
end-policy router bgp 100
 address-family ipv4 unicast
  maximum-paths eibgp 32 equal-cost route-policy EIBGP
!
```

Step 5 Verify that eiBGP multipath and the route policy are working as intended.

Example:

```
Router# show bgp 203.0.113.99/32
BGP routing table entry for 203.0.113.99/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                27          27
Last Modified: Feb 23 16:08:54.000 for 04:12:23
Paths: (7 available, best #2)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.1 0.4
  Path #1: Received by speaker 0
  Not advertised to any peer
  200 300
    209.165.200.11  from 209.165.200.11 (192.168.0.3), -> From R4
   Origin IGP, localpref 100, valid, external, multipath
      Received Path ID 0, Local Path ID 0, version 0
      Community: 2:1
      Origin-AS validity: (disabled)

  Path #2: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.1 0.4
  200 300
    209.165.201.1  from 209.165.201.1  (209.165.201.1) -> From R4
   Origin IGP, localpref 100, valid, external, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 27
      Community: 2:1
      Origin-AS validity: (disabled)

  Path #3: Received by speaker 0
  Not advertised to any peer
  200 300, (Received from a RR-client)
    192.168.2.6 (metric 2) from 198.51.100.1  (198.51.100.1)  -> From R3
   Origin IGP, localpref 100, valid, internal, multipath, backup, add-path
      Received Path ID 0, Local Path ID 2, version 6
      Community: 2:1
  Path #4: Received by speaker 0
  Not advertised to any peer
  200 300, (Received from a RR-client)
    192.168.0.6 (metric 2) from 192.0.2.1 (192.0.2.1)  -> From R5
      Origin IGP, localpref 100, valid, internal
      Received Path ID 0, Local Path ID 0, version 0
      Community: 11:11 99:99

  Path #5: Received by speaker 0
  Not advertised to any peer
  200 300, (Received from a RR-client)
    192.168.0.2 (metric 5) from 192.168.0.2 (192.168.0.2)  -> From R2
      Origin IGP, localpref 100, valid, internal
      Received Path ID 0, Local Path ID 0, version 0
      Community: 2:1 99:99
/* The router does not select Path 5, even though it satisfies the route-policy community constraint,
 because it has a higher metric (i.e., metric 5) than the best path of its path type (i.e., iBGP
metric 2). */

  Path #6: Received by speaker 0
  Not advertised to any peer
  200 300, (Received from a RR-client)
    192.168.0.4 (metric 2) from 192.168.0.4 (192.168.0.4) -> From R5
      Origin IGP, localpref 100, valid, internal
      Received Path ID 0, Local Path ID 0, version 0
      Community: 11:11 99:99
```

```
Path #7: Received by speaker 0
Not advertised to any peer
100 300, (Received from a RR-client)
  192.168.0.5 (metric 2) from 192.168.0.5 (192.168.0.5) -> From R3
 Origin IGP, localpref 100, valid, internal, multipath
    Received Path ID 0, Local Path ID 0, version 0
    Community: 2:1
```

Review the output to confirm that multiple eligible paths are installed and used according to the route policy and ECMP configuration.

The router selects paths for multipath if they match the community criteria and the metric of the best path within the same path type, iBGP or eBGP. A path with a higher metric than the best path of its type is not selected, even if it meets the community constraint.

The router uses ECMP for eiBGP routes that meet your route policy conditions, improving traffic distribution and load balancing.

Resilient hashing and flow auto-recovery

Resilient hashing and flow auto-recovery is a network reliability feature that

- maintains stable traffic flows during Equal-Cost Multipath (ECMP) path failures by only rerouting traffic affected by the failed path,
- prevents unnecessary rebalancing of existing flows to new links, and
- automatically restores original flow distribution when a failed path or server returns to service.
- ECMP: A routing technique that balances traffic across multiple equal-cost paths to the same destination.
- Bucket: A logical mapping of flows to paths in a hashing algorithm.

Table 20: Feature History Table

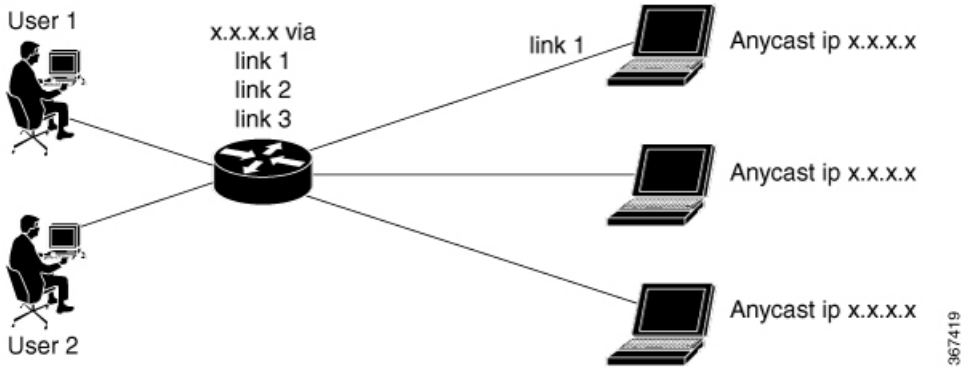
Feature Name	Release Name	Description
Resilient hashing and flow auto-recovery	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100])(select variants only*). You can ensure no packet loss and optimal load distribution across available paths by automatically rerouting data flows during link failures. This feature enhances network reliability by maintaining continuous service and dynamically adjusting to network topology changes without manual intervention. It seamlessly integrates with existing configurations, offering high availability and reducing downtime, thus keeping network operations uninterrupted and efficient. *Previously this feature was supported on Q200 and Q100. It is now extended to Cisco 8712-MOD-M routers.

Resilient hashing and flow auto-recovery let you selectively override the default equal cost multipath (ECMP) behavior during an ECMP path failure. This feature redirects only the flows on failed links and prevents all existing flows from being rehashed to a new link. It also allows a recovered link or server to be reused for sessions when it becomes available.

Impact of ECMP path failures on traffic flows

Prior to the implementation of resilient hashing and flow auto-recovery feature, ECMP load balances traffic across all available paths to a destination. If one path fails, ECMP rehashes the traffic and selects new next hops for each flow.

Figure 13: ECMP Path Failure



For example, if you have three links—link 1, link 2, and link 3—a traffic flow that originally uses link 1 may switch to link 3 after a failure, even if only link 2 fails.

This redistribution of traffic flows does not cause issues in traditional core networks because end-to-end connectivity is maintained and users are not affected. However, in data center environments, load balancing caused by this redistribution can create problems.

In data centers where multiple servers connect through ECMP, rehashing may cause active flows to move, which can reset TCP sessions and disrupt applications.

Benefits of resilient hashing and flow auto-recovery

- Maintains uninterrupted network operations and high availability
- Minimizes traffic disruption and session resets during link failures or recoveries, and
- Supports dynamic adjustment to topology changes without manual intervention.

How resilient hashing and flow auto-recovery work

Summary

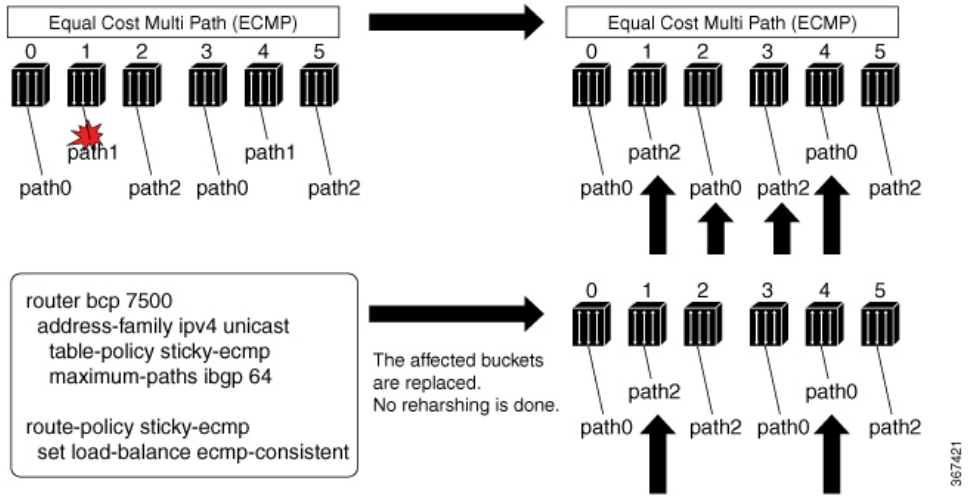
The key components involved in the process are:

- Resilient hashing configuration: Determines how flows are distributed and reassigned when a path fails.
- Route policy language (RPL): Used to specify the prefixes that require resilient hashing and flow auto-recovery.
- ECMP path list: Represents the set of available equal-cost paths for a given prefix.

Resilient hashing and flow auto-recovery help maintain consistent traffic distribution and minimize disruption during ECMP path failures and recoveries. This process ensures only affected traffic flows are redirected, while existing flows remain stable.

Workflow

Figure 14: Resilient hashing and flow auto-recovery



These stages describe how resilient hashing and flow auto-recovery work.

1. The router uses an RPL to define prefixes with associated ECMP path lists, such as path 0, path 1, and path 2 for prefix X.
2. If a path fails, for example, path 1:
   - Without resilient hashing: The router performs a full rehashing, redistributing all flows across the remaining available paths, for example, path 0, path 2, and path 0.
   - With resilient hashing and flow auto-recovery enabled: Only the affected traffic buckets are reassigned, for example, the new path list becomes path 0, path 0, and path 2, and no complete rehash occurs.
3. When the failed path becomes active again, for example, path 1:
   - Without resilient hashing and flow auto-recovery: The path is not reused until one of the following actions happens:
     - Addition of a new path to ECMP
     - Use of the **clear route** command
     - Removal and reapplication of a table policy followed by a **commit**, or
     - Configuration of the **cef consistent-hashing auto-recovery** command
   - With resilient hashing and flow auto-recovery enabled: Sessions previously moved to other paths during the failure are automatically rehashed back to the restored path. Only these specific sessions are disrupted, while others remain unaffected.

Result

Resilient hashing and flow auto-recovery provide stable traffic flows during ECMP path failures and recoveries, minimizing service disruption and ensuring efficient use of all available paths.

Configure resilient hashing and flow auto-recovery

To realize resilient hashing and flow auto-recovery, you can use persistent load balancing, also known as sticky ECMP. Sticky ECMP ensures that when a path failure occurs, only the flows that relied on the failed path are reassigned, while all other flows continue on their original routes.

Traditional ECMP load balances traffic across multiple available paths. When a path fails, ECMP redistributes all flows, which can disrupt established sessions. Persistent load balancing, also known as sticky ECMP, ensures that only flows using the failed path are reassigned, while all other flows remain unchanged.

Before you begin

- Make sure BGP and ECMP are already configured in your network.
- Identify the prefixes that require sticky ECMP.

Follow these steps to configure resilient hashing and flow auto-recovery using sticky ECMP.

Procedure

Step 1 Enter BGP configuration mode and set up the table policy for sticky ECMP.

Example:

```
Router(config)# router bgp 7500
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy sticky-ecmp
Router(config-bgp-af)# bgp attribute-download
Router(config-bgp-af)# maximum-paths ebgp 64
Router(config-bgp-af)# maximum-paths ibgp 32
Router(config-bgp-af)# exit
Router(config-bgp)# exit
```

Step 2 Define a route policy that applies sticky ECMP to the required destination prefix.

Example:

```
Router(config)# route-policy sticky-ecmp
Router(config-rpl)# if destination in (192.1.1.1/24) then
Router(config-rpl-if)# set load-balance ecmp-consistent
Router(config-rpl-if)# else
Router(config-rpl-else)# pass
Router(config-rpl-else)# endif
RP/0/0/CPU0:ios(config-rpl)# end-policy
RP/0/0/CPU0:ios(config)#
```

Step 3 Enable automatic recovery to restore the original hashing state after failed paths become active.

Example:

```
Router(config)# cef consistent-hashing auto-recovery
```

Step 4 Clear the route to recover the original hashing state after failed paths come back up and avoid affecting new flows.

Example:

```
Router(config)# clear route 192.0.2.0/24
```

Step 5 Verify the running configuration.

Example:

```
Router#show running-config
router bgp 7500
 address-family ipv4 unicast
  table-policy sticky-ecmp
  bgp attribute-download
  maximum-paths ebgp 64
  maximum-paths ibgp 32


cef consistent-hashing auto-recovery

clear route 192.0.2.0/24
```

Step 6 Verify that persistent load balancing is working as expected by checking the path distribution before and after a link failure.

a) Run the **show cef** command to check the path distribution before a link failure.

Example:

```
Router# show cef 192.0.2.0/24
 LDI Update time Sep  5 11:22:38.201
   via 10.1.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 0 NHID 0x0 [0x57ac4e74 0x0]
    next hop 10.1.0.1/32 via 10.1.0.1/32
   via 10.2.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 1 NHID 0x0 [0x57ac4a74 0x0]
    next hop 10.2.0.1/32 via 10.2.0.1/32
   via 10.3.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 2 NHID 0x0 [0x57ac4f74 0x0]
    next hop 10.3.0.1/32 via 10.3.0.1/32


    Load distribution (consistent): 0 1 2 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   GigabitEthernet0/0/0/0    10.1.0.1
    1     Y   GigabitEthernet0/0/0/1    10.2.0.1
    2     Y   GigabitEthernet0/0/0/2    10.3.0.1
```

b) Run the **show cef** command to check the path distribution after a link failure.

Example:

```
Router# show cef 192.0.2.0/24
 LDI Update time Sep  5 11:23:13.434
   via 10.1.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 0 NHID 0x0 [0x57ac4e74 0x0]
    next hop 10.1.0.1/32 via 10.1.0.1/32
   via 10.3.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 1 NHID 0x0 [0x57ac4f74 0x0]
    next hop 10.3.0.1/32 via 10.3.0.1/32

    Load distribution (consistent) : 0 1 2 (refcount 1)
    Hash  OK  Interface                 Address
    0     Y   GigabitEthernet0/0/0/0    10.1.0.1
 1*    Y   GigabitEthernet0/0/0/0    10.1.0.1
    2     Y   GigabitEthernet0/0/0/2    10.3.0.1
```

The reassignment of bucket 1 to GigabitEthernet 0/0/0/0, indicated by the "*" symbol, shows that this path is being used as a replacement for a failed path.

Persistent load balancing ensures that only flows on failed paths are reassigned, maintaining session stability for all other traffic.

BGP selective multipath

BGP selective multipath is a routing feature that

- allows you to install multiple parallel paths to the same destination in the routing table
- applies the multipath function only to specified BGP peers instead of all configured peers, and
- enables fine-grained control over which neighbor routes are eligible for multipath installation.

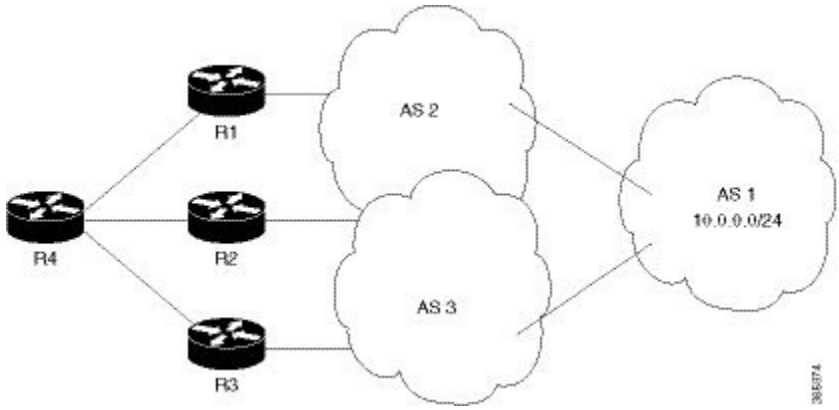
Traditional BGP multipath installs multiple paths from all configured peers by default. With selective multipath, you choose specific neighbors that participate in multipath routing.

Guidelines for BGP selective multipath

- Always include the best path in the set of multipaths; BGP selective multipath does not affect best path calculations.
- Ensure provider edge (PE) paths for VPN prefixes are treated as eligible multipaths.
- Use the **next-hop-unchanged multipath** command to avoid overwriting next-hop information before advertising multipaths.
- Configure the multipath option on neighbors that should be eligible for parallel path installation in the routing table.
- Set the maximum number of selective multipaths for iBGP and eBGP neighbors by using the **maximum-paths ... selective** command.

Topology for BGP selective multipath

Figure 15: Topology for BGP selective multipath



In this sample topology, router R4 receives parallel paths to the same destination from routers R1, R2, and R3. If only R1 and R2 are configured as selective multipath neighbors on R4, then only the parallel paths from R1 and R2 are installed in R4 routing table. Routes from R3 are not used for multipath unless configured.

Configure BGP selective multipath

Enable BGP selective multipath so that only selected neighbors provide multipath routes.

Use this task to control which BGP neighbors contribute multipath routes in your network, improving path management and stability.

Before you begin

Configure your network topology with iBGP or eBGP before enabling selective multipath.

- Make sure iBGP or eBGP are already running on your routers.
- Design your network topology and identify neighbor IP addresses for selective multipath.

Follow these steps to configure BGP selective multipath on router R4.

Procedure

Step 1 Enter BGP address-family configuration and set maximum selective multipaths.

a) Configure selective multipath for iBGP and eBGP.

Example:

```
Router(config)# router bgp 1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# maximum-paths ibgp 4 selective
Router(config-bgp-af)# maximum-paths ebgp 5 selective
Router(config-bgp-af)# commit
```

b) Configure selective multipath for eiBGP.

Example:

```
Router(config)# router bgp 1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# maximum-paths eibgp 6 selective
Router(config-bgp-af)# commit
```

Choose the right configuration based on your network topology.

Step 2 Configure neighbors for selective multipath.

a) For neighbors eligible for multipath, R1 and R2:

Example:

```
Router(config-bgp)# neighbor 1.1.1.1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# multipath
Router(config-bgp-nbr-af)# commit

Router(config-bgp-nbr)# neighbor 2.2.2.2
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# multipath
Router(config-bgp-nbr-af)# commit
```

b) For a neighbor not eligible for multipath, R3:

Example:

```
Router(config-bgp-nbr)# neighbor 3.3.3.3
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# commit
```

Only the specified neighbors, R1 and R2, provide parallel multipath routes in the routing table of router R4.

iBGP multipath load sharing

The iBGP multipath load sharing is a BGP capability that:

- allows a router to install multiple iBGP paths to the same destination in the routing table,
- enables load sharing across these eligible paths, and
- improves network efficiency by leveraging path diversity.

By default, when a BGP router without a local policy receives multiple iBGP routes to the same destination, it selects only one best path and installs it in the routing table. With the iBGP multipath load sharing feature, the router can install multiple iBGP paths as best paths for the same destination, allowing traffic to be shared across these paths.

Benefits of iBGP multipath load sharing

These are some of the benefits that iBGP multipath load sharing provides:

- Increases bandwidth utilization by distributing traffic over multiple iBGP paths

- Improves network resiliency and redundancy
- Enhances path diversity, reducing the risk of single points of failure, and
- Optimizes traffic flow and balances network load.

Restrictions for iBGP multipath load sharing

- You cannot use per-VRF label mode with Carrier Supporting Carrier (CSC) networks that have both internal and external BGP multipath.
- Do not use per-VRF label mode for BGP PIC edge with eBGP multipath, because it may cause routing loops. Use per-prefix label mode instead.
- Per-VRF label mode does not support BGP best external within the same address family, as it can cause routing loops during network re-convergence.

Configure iBGP multipath load sharing

Enable iBGP multipath load sharing to use multiple iBGP paths for the same destination and improve load distribution.

Use this task to configure the maximum number of iBGP paths for load sharing in the BGP routing process.

Before you begin

- Confirm you have the correct autonomous system (AS) number for your router.
- Determine the required address family, IPv4 or IPv6, unicast or multicast.
- Decide the maximum number of iBGP paths you want to configure.

Procedure

Step 1 Configure the AS number for your router.

Example:

```
Router# config
Router(config)# router bgp 100
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 2 Run the **address-family** {**ipv4**|**ipv6**} {**unicast**|**multicast**} command to enter the appropriate address family submode.

Example:

```
Router(config-bgp)# address-family ipv4 multicast
```

Step 3 Run the **maximum-paths ibgp** *number* command to configures the maximum number of iBGP paths for load sharing.

Example:

```
Router(config-bgp-af)# maximum-paths ibgp 30
Router(config-bgp-af)# commit
```

Step 4 Verify the running configuration.

Example:

```
Router# show running-config
router bgp 100
 address-family ipv4 multicast
  maximum-paths ibgp 30
 !
!
end
```

This sample configuration uses 30 iBGP paths for load sharing.

BGP multipath next-hop and as-path control

BGP multipath next-hop and as-path control is a set of enhancements that:

- prevents overwriting of **next-hop** values for multipath prefixes,
- allows you to control how **as-path** information is used when selecting multipath routes, and
- helps maintain predictable and loop-free routing behavior.

You can use the **next-hop-unchanged multipath** command to ensure that BGP does not recalculate the next hop for multipath prefixes.

The **bgp multipath as-path ignore onwards** command enables you to ignore **as-path** attributes beyond a certain point when computing multipath, allowing for more flexible path selection.

Guidelines for BGP multipath next-hop and as-path control

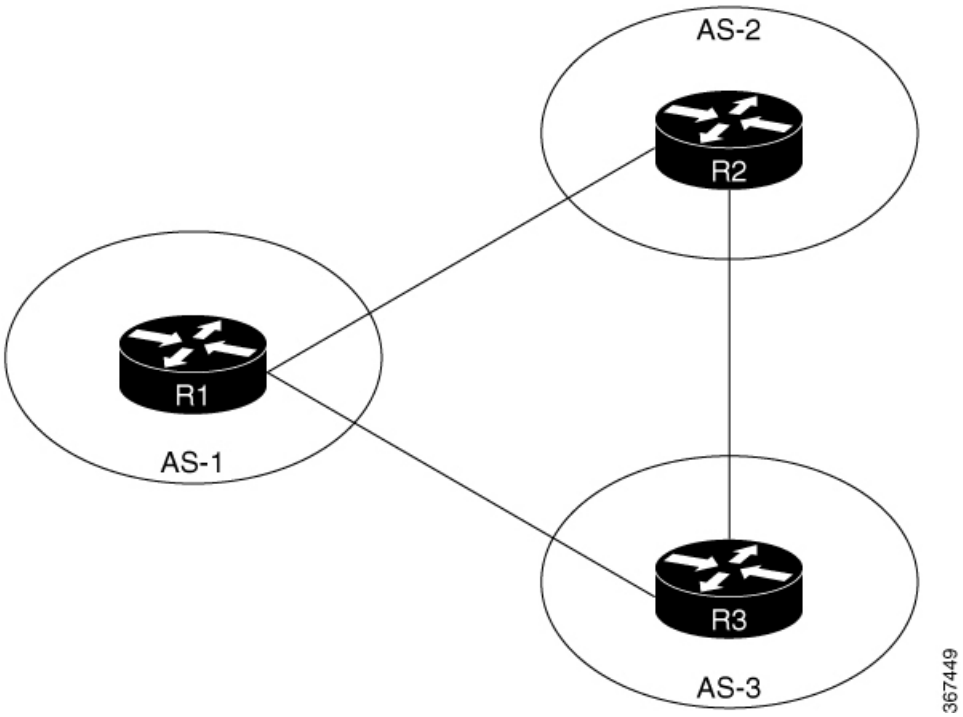
Use the **next-hop-unchanged multipath** command to disable next-hop recalculation for multipath prefixes and preserve the original next-hop values.

Caution Use the **bgp multipath as-path ignore onwards** command with caution. If you enable this command on multiple interconnected routers, it may cause routing loops.

Topology illustrating routing loop formation

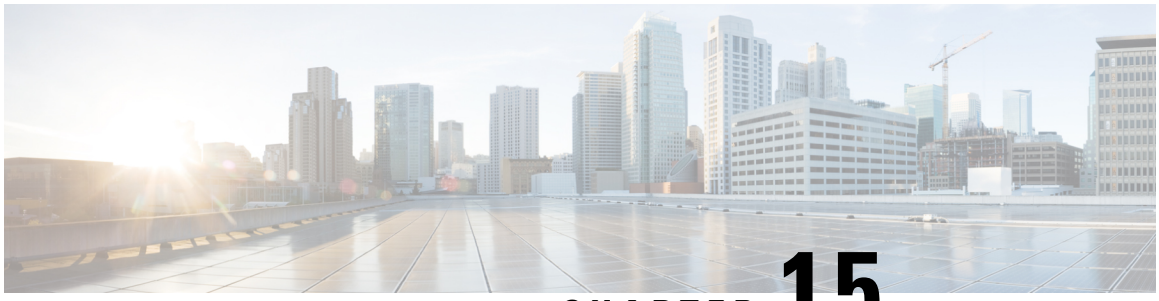
Illustrate how specific BGP multipath configurations can create routing loops between connected routers.

Figure 16: Topology example: routing loop formation



Consider three routers named R1 in AS-1, R2 in AS-2, and R3 in AS-3, with each router in a different autonomous system and all interconnected. R1 announces a prefix to both R2 and R3. If R2 and R3 are both configured for multipath and have the **bgp multipath as-path ignore onwards** command enabled, each router may forward part of its traffic to the other. This behavior creates a forwarding loop between R2 and R3. To avoid such loops, do not enable the **as-path ignore onwards** option on interconnected routers.

Topology illustrating routing loop formation



CHAPTER 15

High-scale BGP Multipath and Load Balancing

This chapter describes high-scale BGP multipath and load balancing techniques to improve network scalability and efficiency.

High-scale BGP multipath and load balancing refers to advanced networking features that

- enable routers to distribute traffic across multiple paths to a destination
- support configurations for various Equal Cost Multipath (ECMP) levels, including 64-way, 128-way, and 256-way ECMP
- introduce hierarchical load balancing for even greater scalability, supporting up to 1,024 ECMPs or 256 Unequal Cost Multipath (UCMP) paths, and
- are crucial for high-traffic networks where efficient load balancing is necessary to optimize bandwidth utilization and improve network performance.

You can leverage ECMP with these scaling options:

- 64-way ECMP configures up to 64 equal-cost next hops and load balances traffic over up to 64 LSPs.
- 128-way ECMP allows the routers to configure up to 128 ECMP next hops for BGP (iBGP/eBGP) in IPv4/IPv6 and support 64 IGP multipaths.
- 256-way ECMP configures up to 256 next hops for ECMP in IPv4/IPv6 and optimizes network bandwidth by load balancing traffic across parallel paths.
- Hierarchical load balancing model expands ECMP scalability beyond hardware limitations by splitting routes.

- 64-way multipath ECMP, on page 185
- 128-way multipath ECMP , on page 186
- 256-way multipath ECMP, on page 189
- Hierarchical load balancing, on page 193

64-way multipath ECMP

64-way multipath ECMP is a BGP feature that

- supports configuration of up to 64 equal-cost multipath (ECMP) next hops for BGP, and

- enables overloaded routers to load balance traffic over up to 64 LSPs.

ECMP routing is a routing strategy that

- enables next-hop packet forwarding to a single destination over multiple best paths which tie for top place in routing metric calculations
- can be used with most routing protocols as it is a per-hop decision limited to a single router, and
- potentially offers substantial increases in bandwidth by load-balancing traffic over multiple paths.

128-way multipath ECMP

The 128-way multipath ECMP is a BGP feature that

- enables the router to support up to 128 BGP and 64 Interior Gateway Protocol (IGP) parallel multipaths to a destination
- enables you to configure up to 128 ECMP next hops for BGP in IPv4 and IPv6 on Cisco 8000 Series Routers, and
- allows up to 128-path ECMP in the global table for BGP, specifically for iBGP and eBGP prefixes.

Table 21: Feature History Table

Feature Name	Release Name	Description
128-way ECMP	Release 7.3.1	This feature enables the router to support up to 128 BGP and 64 IGP parallel multipath routes to a destination.

Restrictions for 128-way multipath ECMP

These restrictions apply to the 128-way multipath ECMP feature:

- Virtual Routing and Forwarding (VRF) and global routing tables in IGP support 64-path ECMP as the maximum for paths, with up to 128 paths supported when backup is configured.
- BGP and IGP in UCMP support 64-path ECMP.
- Label Distribution Protocol (LDP) receives support for only 64-way ECMP from the Routing Information Base (RIB). As a result, Label Switch Database (LSD) receives a maximum of 64 equal-cost multipaths from all clients.

Configure 128-multipath ECMP

Configure 128-multipath ECMP for iBGP, eBGP, or eiBGP to enable traffic load balancing across multiple paths.

You can configure eBGP and iBGP multipath to function together. However, if eiBGP multipath is already configured, you cannot configure iBGP or eBGP multipath.

Procedure

Step 1 Configure 128-multipath ECMP for the desired BGP type, iBGP, eBGP, or eiBGP.

- Configure 128-multipath ECMP in iBGP.

```
Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 10.10.10.11
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#maximum-paths ibgp 128
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#maximum-paths ibgp 128
Router(config-bgp-af)#commit
```

Running configuration:

```
Router# show run router bgp
router bgp 100
bgp router-id 10.10.10.11 address-family ipv4 unicast
maximum-paths ibgp 128
!
address-family ipv6 unicast maximum-paths ibgp 128
```

- Configure 128-multipath ECMP in eBGP.

```
Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 10.10.10.11
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#maximum-paths ebgp 128
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#maximum-paths ebgp 128
Router(config-bgp-af)#commit
```

Running configuration:

```
Router# show run router bgp
router bgp 100
bgp router-id 10.10.10.11 address-family ipv4 unicast
maximum-paths ebgp 128
!
address-family ipv6 unicast maximum-paths ebgp 128
```

- Configure 128-multipath ECMP in eiBGP.

```
Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 10.10.10.11
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#maximum-paths eibgp 128
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#maximum-paths eibgp 128
Router(config-bgp-af)#commit
```

Running configuration:

```
Router# show run router bgp
router bgp 100
bgp router-id 10.10.10.11 address-family ipv4 unicast
maximum-paths eibgp 128
```

```
 !
 address-family ipv6 unicast maximum-paths eibgp 128
```

Step 2 Verify the BGP multipath marking in the routing table, FIB, and RIB, and confirm the total number of IPv4 and IPv6 ECMP paths, as well as the platform's ECMP capabilities.

a) Verify the BGP multipath marking.

Example:

```
Router# show bgp 192.0.2.254/24

BGP routing table entry for 191.1.0.0/24 Versions:
Process bRIB/RIB SendTblVer
Speaker 11008 11008
Last Modified: Aug 14 13:59:39.403 for 00:00:05
Paths: (35 available, best #1)
Advertised IPv4 Unicast paths to peers (in unique update groups): 100.101.3.2
Path #1: Received by speaker 0
Advertised IPv4 Unicast paths to peers (in unique update groups): 100.101.3.2
Local, (received & used)
120.0.101.1 from 120.0.101.1 (120.0.101.1)
Origin IGP, localpref 100, valid, internal, best, group-best, multipath Received Path ID 0, Local
 Path ID 1, version 7708
Path #2: Received by speaker 0
Advertised IPv4 Unicast paths to peers (in unique update groups): 100.101.3.2
Local, (received & used)
120.0.102.1 from 120.0.102.1 (120.0.102.1)
Origin IGP, localpref 100, valid, internal, multipath Received Path ID 0, Local Path ID 6, version
 11008
........
Path #128: Received by speaker 0
Advertised IPv4 Unicast paths to peers (in unique update groups): 100.101.3.2
Local, (received & used)


120.0.227.1 from 120.0.227.1 (120.0.227.1)
Origin IGP, localpref 100, valid, internal, multipath Received Path ID 0, Local Path ID 6, version
 14008
```

b) Verify the BGP multipath marking in FIB.

Example:

```
Router# show cef 192.0.2.254/24
191.1.0.0/24, version 46115, internal 0x5000001 0x40 (ptr 0xd236928) [1], 0x0 (0xe715668), 0x0
(0x0)
Updated Aug 14 13:59:39.007
Prefix Len 24, traffic index 0, precedence n/a, priority 4
via 120.0.101.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080] path-idx 0 NHID 0x0
[0xd236a00 0x0]
next hop 120.0.101.1/32 via 120.0.101.1/32
via 120.0.102.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080] path-idx 1 NHID 0x0
[0xde9a6d0 0x0]
next hop 120.0.102.1/32 via 120.0.102.1/32
via 120.0.103.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080] path-idx 2 NHID 0x0
[0xde9a010 0x0]
….
via 120.0.227.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080] path-idx 127 NHID 0x0
 [0xde9a010 0x0]
```

c) Verify the BGP multipath marking in RIB.

Example:

```
Router# show route 192.0.2.254/24

Routing entry for 191.1.0.0/24
Known via "bgp 1", distance 200, metric 0, type internal Installed Aug 14 13:59:38.971 for 00:11:48
Routing Descriptor Blocks
120.0.101.1, from 120.0.101.1, BGP multi path Route metric is 0
120.0.102.1, from 120.0.102.1, BGP multi path Route metric is 0
…
120.0.227.1, from 120.0.227.1, BGP multi path Route metric is 0
```

d) Verify the total number of IPv4 ECMP paths.

Example:

```
Router# show bgp ipv4 unicast 191.1.0.0/24 | i multipath | utility wc -l
128
```

e) Verify the total number of IPv6 ECMP paths.

Example:

```
Router# show bgp ipv6 unicast 191:1::/64 | i multipath | utility wc -l
128
```

f) Verify the IPv4 128-multipath ECMP.

Example:

```
Router# show route ipv4 191.1.0.0/24 | i multipath | utility wc -l
128
Router# show cef ipv4 191.1.0.0/24 | i multipath | utility wc -l
128
```

g) Verify the IPv6 128-multipath ECMP.

Example:

```
Router# show route ipv6 191:1::/64 | i multi | utility wc -l
128
Router# show cef ipv6 191:1::/64 | i multipath | utility wc -l
128
```

h) Verify the ECMP capabilities of the platform.

Example:

```
Router# show cef misc
Platform capabilities:

L3 loadbalancing levels: 2 L3 Hash buckets: 64
L3 recursive Hash buckets: 128 L3 Unequal cost hash buckets: 64
```

256-way multipath ECMP

256-way multipath ECMP is a BGP feature that

- enables configuration of up to 256 next hops for ECMP in BGP in both IPv4 and IPv6, and
- optimizes network bandwidth by load balancing traffic across parallel paths.

Equal-cost multipath routing (ECMP) is a network technique that allows data packets destined for a single location to be sent via several optimal routes. These routes are considered equally efficient based on routing criteria. This approach works with most network protocols because each router independently determines the best forwarding path. ECMP can enhance network capacity by distributing data flow across diverse routes. With the 256-path ECMP capability, a router can use up to 256 simultaneous connections to reach a given endpoint.

Cisco 8000 Series Routers allow configuration of up to 256 ECMP next hops for BGP in both IPv4 and IPv6. This maximum 256-path ECMP capability extends to the global BGP table, particularly for iBGP and eBGP prefixes.

Table 22: Feature History Table

Feature Name	Release Name	Description
256-way ECMP	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.
256-way ECMP	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.
256-way ECMP	Release 24.4.1	Introduced in this release on: Fixed Systems (8200, 8700); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can now configure up to 256 ECMP next hops for BGP in both IPv4 and IPv6. This optimizes network bandwidth by load balancing traffic across parallel paths.

Restrictions for 256-way multipath ECMP

These restrictions apply to the 256-way multipath ECMP feature:

- BGP and IGP in UCMP support 256 path ECMP for IP, and 64 path ECMP for MPLS.
- LDP receives only 64 ECMP support from the RIB. As a result, LSD receives a maximum of 64 ECMPs from all clients.

Configure 256-way multipath ECMP

Configure 256-way multipath ECMP for iBGP, eBGP, or eiBGP to enable traffic load balancing across multiple paths.

You can configure eBGP and iBGP multipath to function together. However, if eiBGP multipath is already configured, you cannot configure iBGP or eBGP multipath.

Procedure

Step 1 Configure 256-multipath ECMP for the desired BGP type, iBGP, eBGP, or eiBGP.

a) Configure 256-multipath ECMP in iBGP.

Example:

```
Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 10.10.10.11
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#maximum-paths ibgp 256
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#maximum-paths ibgp 256
Router(config-bgp-af)#commit
```

b) Configure 256-multipath ECMP in eBGP.

Example:

```
Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 10.10.10.11
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#maximum-paths ebgp 256
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#maximum-paths ebgp 256
Router(config-bgp-af)#commit
```

c) Configure 256-multipath ECMP in eiBGP.

Example:

```
Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 10.10.10.11
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#maximum-paths eibgp 256
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#maximum-paths eibgp 256
Router(config-bgp-af)#commit
```

Step 2 Verify the BGP multipath marking.

Example:

```
Router# show bgp 10.0.2.254/24

BGP routing table entry for 10.1.0.0/24
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker              11008       11008
Last Modified: Aug 14 13:59:39.403 for 00:00:05
Paths: (35 available, best #1)
  Advertised IPv4 Unicast paths to peers (in unique update groups):
```

```
    10.101.3.2
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    10.101.3.2
  Local, (received & used)
    10.0.101.1 from 10.0.101.1 (10.0.101.1)
      Origin IGP, localpref 100, valid, internal, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 7708
  Path #2: Received by speaker 0
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    10.101.3.2
  Local, (received & used)
    10.0.102.1 from 10.0.102.1 (10.0.102.1)
      Origin IGP, localpref 100, valid, internal, multipath
      Received Path ID 0, Local Path ID 6, version 11008
........
Path #256: Received by speaker 0
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    10.101.3.2
  Local, (received & used)
    10.0.227.1 from 10.0.227.1 (10.0.227.1)
      Origin IGP, localpref 100, valid, internal, multipath
      Received Path ID 0, Local Path ID 6, version 14008
```

Step 3 Verify the BGP multipath marking in FIB.

Example:

```
Router# show cef 10.0.2.254/24

10.1.0.0/24, version 46115, internal 0x5000001 0x40 (ptr 0xd236928) [1], 0x0 (0xe715668), 0x0 (0x0)
 Updated Aug 14 13:59:39.007
 Prefix Len 24, traffic index 0, precedence n/a, priority 4
   via 10.0.101.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 0 NHID 0x0 [0xd236a00 0x0]
    next hop 120.0.101.1/32 via 120.0.101.1/32
   via 10.0.102.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 1 NHID 0x0 [0xde9a6d0 0x0]
    next hop 10.0.102.1/32 via 10.0.102.1/32
   via 10.0.103.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 2 NHID 0x0 [0xde9a010 0x0]
   ….
 via 10.0.227.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 255 NHID 0x0 [0xde9a010 0x0]
```

Step 4 Verify the BGP multipath marking in RIB.

Example:

```
Router# show route 10.0.2.254/24

Routing entry for 10.1.0.0/24
  Known via "bgp 1", distance 200, metric 0, type internal
  Installed Aug 14 13:59:38.971 for 00:11:48
  Routing Descriptor Blocks
    10.0.101.1, from 10.0.101.1, BGP multi path
      Route metric is 0
    10.0.102.1, from 10.0.102.1, BGP multi path
      Route metric is 0
    …
    10.0.227.1, from 10.0.227.1, BGP multi path
      Route metric is 0
```

Step 5 Verify the total number of IPv4 and IPv6 ECMP paths.

Example:

```
Router# show bgp ipv4 unicast 10.1.0.0/24 | i multipath | utility wc -l
256

Router# show bgp ipv6 unicast 2001:DB8::1 | i multipath | utility wc -l
256
```

Step 6 Verify the ECMP capabilities of the platform.

Example:

```
Router# show cef misc
Platform capabilities:
-------------------------
L3 loadbalancing levels: 2
L3 Hash buckets: 64
L3 recursive Hash buckets: 256
L3 IP route Hash buckets: 256
L3 Unequal cost hash buckets: 64
L3 IP Unequal cost hash buckets: 256
```

Hierarchical load balancing

Hierarchical load balancing is a network load balancing model that

- allows you to configure up to 1,024 ECMPs or 256 UCMPs to reach a destination
- achieves this by splitting routes into multiple hierarchical ECMPs or UCMPs, and
- enables ECMP to expand beyond hardware limitations, providing greater scalability.

Table 23: Feature History Table

Feature Name	Release Name	Description
Configure 1024 Equal Cost Multi-Path or 256 Unequal Cost Multi-Path using Hierarchical Load Balancing	Release 7.3.3	In earlier releases, you could configure only up to 128 ECMPs. This feature now allows you to configure up to 1,024 ECMPs or 256 UCMPs to reach a destination. You can achieve this by splitting the routes into multiple hierarchical-based ECMPs or UCMPs. This enables ECMP to expand beyond the hardware limitation of only 512 ECMPs, providing greater scalability.

Routes are split into multiple hierarchical routes to expand the capabilities into two levels of hierarchical ECMPs or UCMPs. You can enable the router to support 1,024 ECMPs or 256 UCMPs by using hierarchical-load-balancing model. Forwarding Information Base (FIB) splits routes based on your configuration, such as Autonomous System (AS) number of each path, group size, and your specific scenario.

As a prerequisite, Cisco Express Forwarding (CEF) must be enabled on all participating routers, since load balancing requires CEF. In unequal cost multipath (UCMP) load-balancing, a weight is associated with each next hop and traffic is distributed across the next hops in proportion to their weight. In ECMP, the route to a destination has multiple next hops and traffic is equally distributed.

Support was previously limited to 128 ECMPs. Starting from Release 7.3.3, the support is extended to 1,024 ECMPs or 256 UCMPs. At any given time, you can configure either 1,024 ECMPs or 256 UCMPs.

Restrictions and guidelines for hierarchical load balancing

- The hierarchical-load-balancing model works only for IP paths, which enhances eBGP interface peering scenario.
- This model does not apply to any path that includes an MPLS label or SRv6 information, because such information prevents CEF from converting flat load-balancing to a hierarchical-load-balancing model.
- CEF must be enabled on all participating routers.

How hierarchical load balancing works

Summary

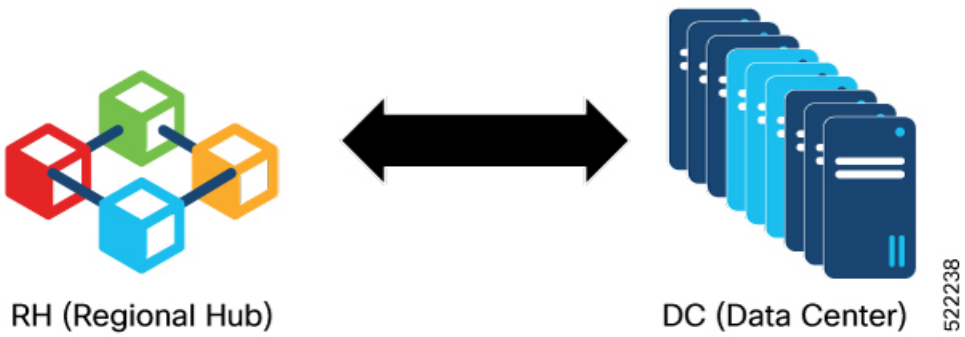
The key components involved in the process are:

- BGP: Downloads neighbor AS information for each path.
- Routing Information Base (RIB): Receives path information from BGP and passes it to FIB.
- FIB: Splits routes and groups paths into hierarchical levels based on configuration and received information.

Hierarchical load balancing is a network load balancing model that extends the capabilities of ECMP and UCMP by splitting and grouping routes into multiple hierarchical levels. This process involves the cooperation of BGP, RIB, and FIB to manage path information and distribute traffic efficiently.

Workflow

Figure 17: Sample topology of hierarchical load balancing



These stages describe hierarchical load balancing.

1. BGP downloads neighbor AS information for each path.

2. BGP forwards this path information to the RIB.
3. The RIB then passes this information to the FIB for regrouping.
4. The FIB splits routes based on the configured group size, AS number of each path, and the specific scenario.
5. The router-specific FIB groups these paths into two hierarchical levels.

Result

The sample topology shows the regional hub connected with 32 data centers with eBGP interface peering. Each data center has 32 interface connections, which adds up to a total of 1,024 ECMPs.

By enabling a hierarchical load-balancing UCMP group size of 256 within a data center, you can scale 256 UCMP eBGP paths on top-of-rack switching using weights.

Configure hierarchical load balancing

Enable and configure hierarchical load balancing to support a higher number of ECMP or UCMP paths for efficient traffic distribution.

Hierarchical load balancing allows you to scale beyond traditional ECMP limits by splitting and grouping routes. To use this feature, enable it and define route policies for UCMP.

Before you begin

Ensure CEF is enabled on all participating routers.

Procedure

Step 1 Run the **cef hierarchical-load-balancing** command to enable hierarchical load balancing with 1024 paths in ECMP mode.

For ECMP mode, the FIB splits or groups paths according to the path attribute of the AS number. Paths to the same destination must be configured on the same AS. Because of hardware limitations, the maximum number of remote AS numbers should be fewer than 128.

- If paths are less than the min-path, FIB uses native forwarding mode.
- If paths are greater than or equal to the min-path, FIB converts the forwarding chain to hierarchical forwarding.

Example:

```
Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 10.10.10.11
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#maximum-paths ebgp 1024
Router(config-bgp-af)#maximum-paths unique-nexthop-check-disable
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#maximum-paths ebgp 1024
Router(config-bgp-af)#maximum-paths unique-nexthop-check-disable
Router(config-bgp-af)#exit
Router(config-bgp)#exit
Router(config)#cef hierarchical-load-balancing ecmp min-paths 128
```

Note
Reload the router so that the **cef hierarchical-load-balancing ecmp min-paths** command takes effect.

Step 2 Configure group size and route policy for UCMP links.

Define an extended community bandwidth routing policy that uses weight values to control traffic on UCMP links.

For example, the system applies the ratio *AS number to Weight number* on the egress of BGP. The FIB uses group size to split or group paths into hierarchical forwarding mode.

- The group size value must be within the hardware limitation of 64 in case of IGP level load balancing.
- The recommended group size is 64 if all paths are ECMP.
- If paths have weight attributes, the recommended group size is only 32.

Example:

```
Router(config)# cef hierarchical-load-balancing ucmp group-size 32

Router(config)#route-policy BW1
Router(config-rpl)#set extcommunity bandwidth (800:10000)
Router(config-rpl)#end-policy
Router(config)#route-policy BW2
Router(config-rpl)#set extcommunity bandwidth (800:20000)
Router(config-rpl)#end-policy
Router(config)#route-policy BW3
Router(config-rpl)#set extcommunity bandwidth (800:40000)
Router(config-rpl)#end-policy
Router(config)#route-policy BW4
Router(config-rpl)#set extcommunity bandwidth (800:80000)
Router(config-rpl)#end-policy
Router(config)#commit
```

Note
Reload the router so that the **cef hierarchical-load-balancing ucmp group-size** command takes effect.

Step 3 Verify CEF configuration for hierarchical load balancing.

a) Verify the BGP IPv4 unicast prefix status.

Example:

```
Router#show bgp ipv4 unicast 1.4.0.0/16
BGP routing table entry for 1.4.0.0/16
Versions:
Process bRIB/RIB SendTblVer
Speaker 3 3
Last Modified: Jan 31 13:14:07.023 for 08:53:38
Paths: (1 available, best #1)
Advertised IPv4 Unicast paths to update-groups (with more than one peer): 0.5
Path #1: Received by speaker 0
Advertised IPv4 Unicast paths to update-groups (with more than one peer): 0.5
Local
100.100.100.200 (metric 30) from 100.100.100.101 (100.100.100.200)
Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best Received Path ID 1,
 Local Path ID 1, version 3
Originator: 100.100.100.200, Cluster list: 100.100.100.101
```

b) Verify the CEF configuration status for ECMP.

Example:

```
Router# show cef 200.1.0.0 detail
200.1.0.0/24, version 92168105, internal 0x5000001 0x40 (ptr 0xaa001058) [1], 0x0 (0x0), 0x0 (0x0)
Updated Sep 8 23:04:13.489
Prefix Len 24, traffic index 0, precedence n/a, priority 4
gateway array (0x9232e858) reference count 596, flags 0x2010, source rib (7), 0 backups
[1 type 3 flags 0x48441 (0xa9df9098) ext 0x0 (0x0)] LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
gateway array update type-time 1 Sep 8 23:04:13.490
LDI Update time Sep 8 23:04:13.490 Weight distribution:
slot 0, weight 5120, normalized_weight 15
slot 1, weight 5120, normalized_weight 15
slot 2, weight 5120, normalized_weight 15
slot 3, weight 5120, normalized_weight 15

Level 1 - Load distribution: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

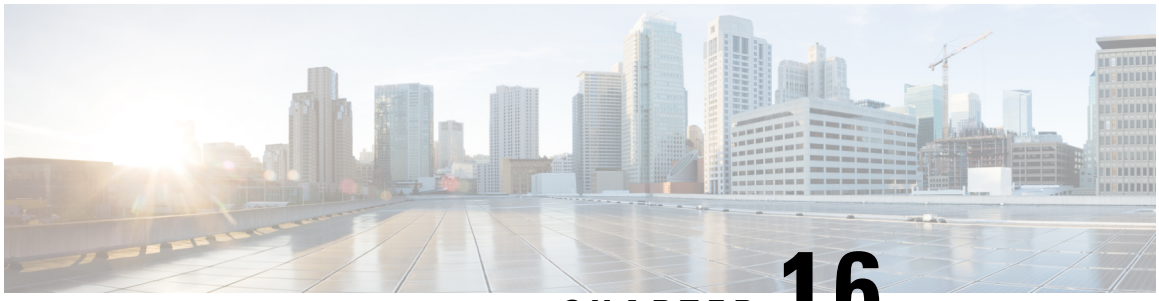
c) Verify the CEF configuration status for UCMP.

Example:

```
Router# show cef 200.1.0.0 detail
200.1.0.0/24, version 6044, internal 0x5000001 0x40 (ptr 0xa64e7738) [1], 0x0 (0x0), 0x0 (0x0)
Prefix Len 24, traffic index 0, precedence n/a, priority 4
gateway array (0x92f14758) reference count 5001, flags 0x2010, source rib (7), 0 backups [1 type
 3 flags 0x40441 (0xa4d331d8) ext 0x0 (0x0)]
LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
gateway array update type-time 3 Nov 26 16:32:49.195
LDI Update time Nov 26 16:32:49.195 Weight distribution:
slot 0, weight 2560, normalized_weight 2
slot 1, weight 2560, normalized_weight 2
slot 2, weight 2560, normalized_weight 2
slot 3, weight 2560, normalized_weight 2
slot 4, weight 1280, normalized_weight 1
slot 5, weight 1280, normalized_weight 1
slot 6, weight 1280, normalized_weight 1
slot 7, weight 1280, normalized_weight 1
Level 1 - Load distribution: 0 0 1 1 2 2 3 3 4 5 6 7
[0] via 241.0.58.138/32, recursive
[1] via 241.0.58.138/32, recursive
[2] via 241.0.58.134/32, recursive
[3] via 241.0.58.134/32, recursive
[4] via 241.0.58.130/32, recursive
[5] via 241.0.58.130/32, recursive
[6] via 241.0.58.126/32, recursive
[7] via 241.0.58.126/32, recursive
[8] via 241.0.58.122/32, recursive
[9] via 241.0.58.118/32, recursive
[10] via 241.0.58.114/32, recursive
[11] via 241.0.58.110/32, recursive
via 241.0.58.138/32, 4 dependencies, recursive [flags 0x0] path-idx 0 NHID 0x0 [0x930b65e0 0x0]
next hop VRF - 'iid', table - 0xe0000203 next hop 241.0.58.138/32 via 241.0.58.138/32
Load distribution: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
27 28 29 30 31 (refcount 1)
Hash OK Interface Address
0 Y TenGigE0/0/0/21/3.1 18.1.0.1
1 Y TenGigE0/0/0/21/3.2 18.1.1.1
2 Y TenGigE0/0/0/21/3.3 18.1.2.1
3 Y TenGigE0/0/0/21/3.4 18.1.3.1
4 Y TenGigE0/0/0/21/3.5 18.1.4.1
5 Y TenGigE0/0/0/21/3.6 18.1.5.1
6 Y TenGigE0/0/0/21/3.7 18.1.6.1
7 Y TenGigE0/0/0/21/3.8 18.1.7.1
8 Y TenGigE0/0/0/21/3.9 18.1.8.1
9 Y TenGigE0/0/0/21/3.10 18.1.9.1
10 Y TenGigE0/0/0/21/3.11 18.1.10.1
```

```
11 Y TenGigE0/0/0/21/3.12 18.1.11.1
12 Y TenGigE0/0/0/21/3.13 18.1.12.1
13 Y TenGigE0/0/0/21/3.14 18.1.13.1
14 Y TenGigE0/0/0/21/3.15 18.1.14.1
15 Y TenGigE0/0/0/21/3.16 18.1.15.1
16 Y TenGigE0/0/0/21/3.17 18.1.16.1
17 Y TenGigE0/0/0/21/3.18 18.1.17.1
18 Y TenGigE0/0/0/21/3.19 18.1.18.1
19 Y TenGigE0/0/0/21/3.20 18.1.19.1
20 Y TenGigE0/0/0/21/3.21 18.1.20.1
21 Y TenGigE0/0/0/21/3.22 18.1.21.1
22 Y TenGigE0/0/0/21/3.23 18.1.22.1
23 Y TenGigE0/0/0/21/3.24 18.1.23.1
24 Y TenGigE0/0/0/21/3.25 18.1.24.1
25 Y TenGigE0/0/0/21/3.26 18.1.25.1
26 Y TenGigE0/0/0/21/3.27 18.1.26.1
27 Y TenGigE0/0/0/21/3.28 18.1.27.1
28 Y TenGigE0/0/0/21/3.29 18.1.28.1
29 Y TenGigE0/0/0/21/3.30 18.1.29.1
30 Y TenGigE0/0/0/21/3.31 18.1.30.1
31 Y TenGigE0/0/0/21/3.32 18.1.31.1
```

Hierarchical load balancing is enabled and configured, allowing for expanded ECMP or UCMP capabilities.



CHAPTER 16

BGP DMZ Bandwidth Management

This chapter describes the various aspects of BGP Demilitarized Zone (DMZ) bandwidth management, including aggregate bandwidth, link bandwidth for unequal cost recursive load balancing, and transitive-bandwidth extended community support.

BGP DMZ aggregate bandwidth

BGP DMZ aggregate bandwidth is a feature that aggregates the link-bandwidth values of DMZ eBGP multipaths when advertising routes to iBGP peers, and enables accurate internal bandwidth representation for better routing decisions.

BGP DMZ aggregate bandwidth operation

BGP aggregates bandwidth without an explicit command if these conditions are met:

- The network has multipaths and all multipaths have link-bandwidth values.
- You set the *next-hop* attribute to *next-hop-self*. The *next-hop* attribute for all routes advertised to the specified neighbor is the address of the local router.
- You do not configure an outbound policy that might change the DMZ link-bandwidth value.

DMZ link bandwidth aggregation rules

DMZ link bandwidth aggregation follows these rules:

- If BGP does not know the DMZ link-bandwidth value (*dmz-link-bandwidth*) for any one of the multipaths (eBGP or iBGP), BGP does not download the DMZ link-bandwidth value for all multipaths, including the best path, to the routing information base (RIB).
- BGP does not consider the DMZ link-bandwidth value of iBGP multipath during aggregation.
- BGP can advertise the route with an aggregate value as a best path or an add-path.

- Add-path does not qualify for DMZ link bandwidth aggregation as the next hop is preserved. BGP does not support configuring next-hop-self for add-path.
- For VPNv4 and VPNv6 address family identifiers (AFIs), if you configure the DMZ link-bandwidth value using an outbound route-policy, specify the route table or use the *additive* keyword. Otherwise, the system does not import routes on the receiving end of the peer.

Configure BGP DMZ aggregate bandwidth

Configure BGP DMZ aggregate bandwidth in a sample topology.

This example uses a topology of R1---(iBGP)---R2---(iBGP)---R3 to demonstrate how aggregated DMZ link-bandwidth values are sent between routers. The routers in the topology advertise and receive aggregated DMZ link-bandwidth values.

- On R1, BGP prefix has:
  - path 1 (bestpath) with link-bandwidth value 100
  - path 2 (eBGP multipath) with link-bandwidth value 30, and
  - path 3 (eBGP multipath) with link-bandwidth value 50.

  When the best path is advertised to R2, R1 sends an aggregated DMZ link-bandwidth value of 180; this is the aggregated value of paths 1, 2, and 3.
- On R2, BGP prefix has:
  - path 1 (bestpath) with link-bandwidth value 60
  - path 2 (eBGP multipath) with link-bandwidth value 200, and
  - path 3 (eBGP multipath) with link-bandwidth value 50.

  When the best path is advertised to R3, R2 sends an aggregated DMZ link-bandwidth value of 310; this is the aggregated value of paths 1, 2, and 3.
- On R3, BGP prefix has:
  - path 1 (bestpath) with LB 180 (learned from R1)
  - path 2 (iBGP multipath) with LB 310 (learned from R2)

This sample configuration demonstrates how to set the link-bandwidth extended community on a per-path basis at either the neighbor-in or neighbor-out policy attach points. The **dmz-link-bandwidth** command is configured under eBGP neighbor configuration mode. All paths received from that particular neighbor are marked with the link-bandwidth extended community when sent to iBGP peers.

Procedure

Step 1 Configure an inbound or outbound route-policy.

Example:

```
Router(config)# extcommunity-set bandwidth dmz_ext
Router(config-ext)# 1:1290400000
Router(config-ext)# end-set
Router(config)#route-policy dmz_rp
Router(config-rpl)#set extcommunity bandwidth dmz_ext
Router(config-rpl)#pass
Router(config-rpl)#end-policy
Router(config)#router bgp 65000
Router(config-bgp)#neighbor 10.0.101.1
Router(config-bgp-nbr)#remote-as 1001
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy dmz_rp in
Router(config-bgp-nbr-af)#route-policy pass out
Router(config-bgp-nbr-af)#commit
```

Step 2 Configure the **dmz-link-bandwidth** command for the BGP neighbor.

Example:

```
Router(config)#router bgp 65000
Router(config-bgp)#neighbor 10.0.101.2
Router(config-bgp-nbr)#remote-as 1001
Router(config-bgp-nbr)#dmz-link-bandwidth
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy pass in
Router(config-bgp-nbr-af)#route-policy pass out
Router(config-bgp-nbr-af)#commit
```

The system applies policy-based link bandwidth settings to BGP neighbors.

Removal of link-bandwidth extended community to iBGP peers

The removal of link-bandwidth extended community to iBGP peers is a feature that:

- minimizes the risk of exposing DMZ link-bandwidth community parameters to network zones where they are not recognized or necessary
- allows BGP to send traffic over multiple internal BGP (iBGP) learned paths, and
- sends traffic proportional to the bandwidth of the links used to exit the autonomous system.

Table 24: Feature History Table

Feature Name	Release Information	Feature Description
Removal of Link-Bandwidth Extended Community to iBGP Peers	Release 7.3.2	The demilitarized zone (DMZ) link-bandwidth extended community allows BGP to send traffic over multiple internal BGP (iBGP) learned paths. The traffic that is sent is proportional to the bandwidth of the links that are used to exit the autonomous system. By default, iBGP propagates DMZ link-bandwidth community. This feature minimizes the risk of exposure of the community parameters, which are used to control the routing policy in the service provider network, to networks zones where they are not recognized or not required.

By default, iBGP propagates DMZ link-bandwidth community. This feature minimizes the risk of exposing community parameters, which control the routing policy in the service provider network, to network zones where they are not recognized or required.

Configure route policy to remove extended communities

Configure a route policy to remove extended communities.

This task provides examples of how to configure route policies to remove extended communities, either all of them or specific ones.

Procedure

Step 1 Choose one of these steps to configure a route policy to remove extended communities

- Configure a route policy to delete all extended communities.

```
Router(config)# route-policy dmz_del_all
Router(config-rpl)# delete extcommunity bandwidth all
Router(config-rpl)# pass
Router(config-rpl)# end-policy
```

- Configure a route policy to delete only the extended communities that match an extended community mentioned in a list.

```
Router(config)# route-policy dmz_CE1_del_non_match
Router(config-rpl)# if destination in (10.9.9.9/32) then
Router(config-rpl-if)# delete extcommunity bandwidth in (10:7000)
Router(config-rpl-if)# endif
Router(config-rpl)# pass
Router(config-rpl)# end-policy
```

- Configure a route policy to delete all extended communities using parameters.

```
Router(config)# route-policy dmz_del_param2($a,$b)
Router(config-rpl)# if destination in (10.9.9.9/32) then
Router(config-rpl-if)# delete extcommunity bandwidth in ($a:$b)
Router(config-rpl-if)# endif
Router(config-rpl)# pass
Router(config-rpl)# end-policy
```

Step 2 Run **show bgp** command to confirm the BGP routing table entries and extended community attributes.

Example:

```
Router# show bgp 10.9.9.9/32

BGP routing table entry for 10.9.9.9/32
Versions:
Process bRIB/RIB SendTblVer
Speaker 15 15
Last Modified: Aug 27 13:06:45.000 for 00:08:21
Paths: (3 available, best #1)
Advertised IPv4 Unicast paths to peers (in unique update groups):
13.13.13.5
Path #1: Received by speaker 0
Advertised IPv4 Unicast paths to peers (in unique update groups):
13.13.13.5
10
```

```
10.10.10.1 from 10.10.10.1 (192.168.0.1)
Origin incomplete, metric 0, localpref 100, valid, external, best, group-best, multipath
Received Path ID 0, Local Path ID 1, version 15
Extended community: LB:10:48
Origin-AS validity: (disabled)
Path #2: Received by speaker 0
Not advertised to any peer
10
11.11.11.3 from 11.11.11.3 (192.168.0.3)
Origin incomplete, metric 0, localpref 100, valid, external, multipath
Received Path ID 0, Local Path ID 0, version 0
Extended community: LB:10:48
Origin-AS validity: (disabled)
Path #3: Received by speaker 0
Not advertised to any peer
10
12.12.12.4 from 12.12.12.4 (192.168.0.4)
Origin incomplete, metric 0, localpref 100, valid, external, multipath
Received Path ID 0, Local Path ID 0, version 0
Extended community: LB:10:48
Origin-AS validity: (disabled)

22:35 30-09-2021
```

BGP DMZ link bandwidth for unequal cost recursive load balancing

BGP DMZ link bandwidth for unequal cost recursive load balancing is a feature that provides support for unequal cost load balancing for recursive prefixes on a local node using BGP DMZ link bandwidth. The system achieves unequal load balancing by using the **dmz-link-bandwidth** command in BGP neighbor configuration mode and the **bandwidth** command in interface configuration mode.

Enable BGP unequal cost recursive load balancing

Enable BGP unequal cost recursive load balancing.

Before you begin

Ensure the network configuration supports BGP and interface bandwidth settings.

Procedure

Step 1 Enter global configuration mode to begin making configuration changes.

Example:

```
Router# config
```

Step 2 Enter BGP router configuration mode and specify the local Autonomous System Number (ASN) for your router.

Example:

```
Router(config)# router bgp 120
```

Step 3 Specify the BGP address family, IPv4 or IPv6 unicast, to configure routing for the desired protocol.

Example:

```
Router(config-bgp)# address-family ipv4 unicast
```

Step 4 Set the maximum number of parallel BGP paths, for eBGP, iBGP, or both, that can be used for load balancing, and optionally enable unequal-cost multipath.

Example:

```
Router(config-bgp-af)# maximum-paths ebgp 3
```

Step 5 Exit the address-family configuration mode and return to BGP router configuration mode.

Example:

```
Router(config-bgp-af)# exit
```

Step 6 Select the BGP neighbor you want to configure by specifying its IP address.

Example:

```
Router(config-bgp)# neighbor 10.0.0.0
```

Step 7 Enable the DMZ link-bandwidth feature for the specified BGP neighbor, allowing the router to use per-link bandwidth information for load balancing.

Example:

```
Router(config-bgp-nbr)# dmz-link-bandwidth
Router(config-bgp-nbr)# commit
```

Configure BGP unequal cost recursive load balancing: example

Provide a sample configuration for unequal cost recursive load balancing.

This example shows the configuration for various interfaces and BGP settings to achieve unequal cost recursive load balancing.

Procedure

Step 1 Execute these commands in configuration mode to set up BGP with multiple neighbors, enable DMZ link-bandwidth, and apply route policies:

Example:

```
Router(config)#interface Loopback0
Router(config-if)#ipv4 address 20.20.20.20 255.255.255.255
Router(config-if)#exit

Router(config)#interface MgmtEth0/RSP0/CPU0/0
Router(config-if)#ipv4 address 8.43.0.10 255.255.255.0
Router(config-if)#exit

Router(config)#interface TenGigE0/3/0/0
Router(config-if)#bandwidth 8000000
Router(config-if)#ipv4 address 11.11.11.11 255.255.255.0
```

```
Router(config-if)#ipv6 address 11:11:0:1::11/64
Router(config-if)#exit

Router(config)#interface TenGigE0/3/0/1
Router(config-if)#bandwidth 7000000
Router(config-if)#ipv4 address 11.11.12.11 255.255.255.0
Router(config-if)#ipv6 address 11:11:0:2::11/64
Router(config-if)#exit

Router(config)#interface TenGigE0/3/0/2
Router(config-if)#bandwidth 6000000
Router(config-if)#ipv4 address 11.11.13.11 255.255.255.0
Router(config-if)#ipv6 address 11:11:0:3::11/64
Router(config-if)#exit

Router(config)#interface TenGigE0/3/0/3
Router(config-if)#bandwidth 5000000
Router(config-if)#ipv4 address 11.11.14.11 255.255.255.0
Router(config-if)#ipv6 address 11:11:0:4::11/64
Router(config-if)#exit

Router(config)#interface TenGigE0/3/0/4
Router(config-if)#bandwidth 4000000
Router(config-if)#ipv4 address 11.11.15.11 255.255.255.0
Router(config-if)#ipv6 address 11:11:0:5::11/64
Router(config-if)#exit

Router(config)#interface TenGigE0/3/0/5
Router(config-if)#bandwidth 3000000
Router(config-if)#ipv4 address 11.11.16.11 255.255.255.0
Router(config-if)#ipv6 address 11:11:0:6::11/64
Router(config-if)#exit

Router(config)#interface TenGigE0/3/0/6
Router(config-if)#bandwidth 2000000
Router(config-if)#ipv4 address 11.11.17.11 255.255.255.0
Router(config-if)#ipv6 address 11:11:0:7::11/64
Router(config-if)#exit

Router(config)#interface TenGigE0/3/0/7
Router(config-if)#bandwidth 1000000
Router(config-if)#ipv4 address 11.11.18.11 255.255.255.0
Router(config-if)#ipv6 address 11:11:0:8::11/64
Router(config-if)#exit

Router(config)#interface TenGigE0/4/0/0
Router(config-if)#description CONNECTED TO IXIA 1/3 transceiver permit pid all
Router(config-if)#exit

Router(config)#interface TenGigE0/4/0/2
Router(config-if)#ipv4 address 9.9.9.9 255.255.0.0
Router(config-if)#ipv6 address 9:9::9/64
Router(config-if)#ipv6 enable
Router(config-if)#exit

Router(config)#route-policy pass-all
Router(config-rpl)#pass
Router(config-rpl)#end-policy

Router(config)#router static
Router(config-static)#address-family ipv4 unicast
Router(config-static-afi)#202.153.144.0/24 8.43.0.1
Router(config-static-afi)#exit
```

```
Router(config-static)#exit

Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 20.20.20.20
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#maximum-paths eibgp 8
Router(config-bgp-af)#redistribute connected
Router(config-bgp-af)#exit

Router(config-bgp)#neighbor 11.11.11.12
Router(config-bgp-nbr)#remote-as 200
Router(config-bgp-nbr)#dmz-link-bandwidth
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy pass-all in
Router(config-bgp-nbr-af)#route-policy pass-all out
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#exit

Router(config-bgp)#neighbor 11.11.12.12
Router(config-bgp-nbr)#remote-as 200
Router(config-bgp-nbr)#dmz-link-bandwidth
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy pass-all in
Router(config-bgp-nbr-af)#route-policy pass-all out
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#exit

Router(config-bgp)#neighbor 11.11.13.12
Router(config-bgp-nbr)#remote-as 200
Router(config-bgp-nbr)#dmz-link-bandwidth
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy pass-all in
Router(config-bgp-nbr-af)#route-policy pass-all out
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#exit

Router(config-bgp)#neighbor 11.11.14.12
Router(config-bgp-nbr)#remote-as 200
Router(config-bgp-nbr)#dmz-link-bandwidth
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy pass-all in
Router(config-bgp-nbr-af)#route-policy pass-all out
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#exit

Router(config-bgp)#neighbor 11.11.15.12
Router(config-bgp-nbr)#remote-as 200
Router(config-bgp-nbr)#dmz-link-bandwidth
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy pass-all in
Router(config-bgp-nbr-af)#route-policy pass-all out
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#exit

Router(config-bgp)#neighbor 11.11.16.12
Router(config-bgp-nbr)#remote-as 200
Router(config-bgp-nbr)#dmz-link-bandwidth
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy pass-all in
Router(config-bgp-nbr-af)#route-policy pass-all out
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#exit
```

```
Router(config-bgp)#neighbor 11.11.17.12
Router(config-bgp-nbr)#remote-as 200
Router(config-bgp-nbr)#dmz-link-bandwidth
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy pass-all in
Router(config-bgp-nbr-af)#route-policy pass-all out
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#exit

Router(config-bgp)#neighbor 11.11.18.12
Router(config-bgp-nbr)#remote-as 200
Router(config-bgp-nbr)#dmz-link-bandwidth
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy pass-all in
Router(config-bgp-nbr-af)#route-policy pass-all out
Router(config-bgp-nbr-af)#exit
Router(config-bgp-nbr)#exit

Router(config-bgp)#exit
Router(config)#commit
```

Step 2 Verify the running configuration.

Example:

```
Router# show running-config
interface Loopback0
 ipv4 address 20.20.20.20 255.255.255.255
!
!
interface FourHundredGige0/1/0/0
 bandwidth 8000000
 ipv4 address 11.11.11.11 255.255.255.0
 ipv6 address 11:11:0:1::11/64
!
interface FourHundredGige0/0/0/0
 bandwidth 7000000
 ipv4 address 11.11.12.11 255.255.255.0
 ipv6 address 11:11:0:2::11/64
!
interface FourHundredGige0/3/0/0
 bandwidth 6000000
 ipv4 address 11.11.13.11 255.255.255.0
 ipv6 address 11:11:0:3::11/64
!
interface FourHundredGige0/4/0/0
 bandwidth 5000000
 ipv4 address 11.11.14.11 255.255.255.0
 ipv6 address 11:11:0:4::11/64
!
interface FourHundredGige0/0/0/0
 bandwidth 4000000
 ipv4 address 11.11.15.11 255.255.255.0
 ipv6 address 11:11:0:5::11/64
!
interface FourHundredGige0/2/0/0
 bandwidth 3000000
 ipv4 address 11.11.16.11 255.255.255.0
 ipv6 address 11:11:0:6::11/64
!
interface FourHundredGige0/3/0/0
 bandwidth 2000000
 ipv4 address 11.11.17.11 255.255.255.0
```

```
 ipv6 address 11:11:0:7::11/64
!
interface FourHundredGige0/3/0/0
 bandwidth 1000000
 ipv4 address 11.11.18.11 255.255.255.0
 ipv6 address 11:11:0:8::11/64
!
interface FourHundredGige0/4/0/0
 description CONNECTED TO IXIA 1/3
 transceiver permit pid all
!
interface FourHundredGige0/4/0/0
 ipv4 address 9.9.9.9 255.255.0.0
 ipv6 address 9:9::9/64
 ipv6 enable
!
route-policy pass-all
  pass
end-policy
!
router static
 address-family ipv4 unicast
  202.153.144.0/24 8.43.0.1
 !
!
router bgp 100
 bgp router-id 10.20.20.20
 address-family ipv4 unicast
  maximum-paths eibgp 8
  redistribute connected
 !
 neighbor 11.11.11.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 11.11.12.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 10.11.13.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 11.11.14.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 11.11.15.12
```

```
 remote-as 200
 dmz-link-bandwidth
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
 !
!
neighbor 11.11.16.12
 remote-as 200
 dmz-link-bandwidth
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
 !
!
neighbor 11.11.17.12
 remote-as 200
 dmz-link-bandwidth
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
 !
!
neighbor 11.11.18.12
 remote-as 200
 dmz-link-bandwidth
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
 !
!
!
```

BGP DMZ link bandwidth enhancement

BGP DMZ link bandwidth enhancement is a BGP feature that

- accurately advertises all valid link bandwidth values
- preserves precision in bandwidth signaling, and
- allows sending and advertising any valid BGP DMZ link bandwidth value, including values less than 1 kilobit per second (kbps).

Table 25: Feature History Table

Feature Name	Release Information	Feature Description
BGP DMZ link bandwidth enhancement	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.

Feature Name	Release Information	Feature Description
BGP DMZ link bandwidth enhancement	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can now send and advertise any valid BGP DMZ link bandwidth value including values below 1 kilobit per second (kbps). Previously, BGP rounded smaller link bandwidth values and failed to advertise values below 1 kbps. The feature introduces these changes: CLI: The **show bgp** command is enhanced to display link bandwidth values in both kbps and bytes per second for improved precision and visibility.

The feature allows you to send and advertise link bandwidth values of any size, overcoming the previous format limitation that restricted values to greater than 1 kbps. With the enhancement, RPL supports 64-bit link bandwidth values.

Benefits of BGP DMZ link bandwidth enhancement

The key benefits of the feature are:

- BGP avoids rounding issues that affected smaller bandwidth values.
- Supports a wider and more granular range of bandwidth values.
- Precise bandwidth advertisement ensures that any valid link bandwidth value is advertised accurately and preserves network performance metrics.

Usage guidelines for BGP DMZ link bandwidth enhancement

- All address families support the feature. However, it is primarily used only with the IPv4 and IPv6 address families that are installed in the RIB by the default and specific VRFs.
- You cannot use DMZ link bandwidth with Multihop eBGP.

Verify DMZ link bandwidth enhancement

Verify the DMZ link bandwidth enhancement.

Before you begin

Follow these steps to verify the DMZ link bandwidth enhancement.

Procedure

Step 1 Run the **show route** command to verify the status of specific routes in the routing table.

Example:

```
Router#show route 209.165.201.1/27
...
Routing entry for 209.165.201.1/27
  Known via "bgp 1", distance 200, metric 0
  Tag 2, type internal
  Installed Mar  6 09:14:55.055 for 00:06:10
  Routing Descriptor Blocks
    1.1.1.1, from 1.1.1.1, BGP multi path
      Route metric is 0, Wt is 16000
    1.1.1.2, from 1.1.1.2, BGP multi path
      Route metric is 0, Wt is 8000
...
```

In the sample output, BGP utilizes link bandwidth values for multipath load balancing and adjusts and distributes weights across different paths.

Step 2 Run the **show cef** command to verify load sharing based on weights derived from link bandwidth values.

Example:

```
Router#show cef 209.165.201.1/27
...
    Weight distribution:
    slot 0, weight 16000, normalized_weight 2
    slot 1, weight 8000, normalized_weight 1

  Level 1 - Load distribution: 0 0 1
  [0] via 1.1.1.1/32, recursive
  [1] via 1.1.1.1/32, recursive
  [2] via 1.1.1.2/32, recursive
...
```

In the sample output, the load distribution levels indicate how packets are distributed among various paths. The weight distribution values indicate how weights are normalized across various slots.

Step 3 Run the **show bgp** command to verify the link bandwidth in bytes per second.

Example:

```
Router#show bgp 209.165.201.1/27
Thu Mar  6 09:22:09.823 UTC
BGP routing table entry for 209.165.201.1/27
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker             160385       160385
Last Modified: Mar  6 09:14:55.098 for 00:07:14
Paths: (2 available, best #1)
```

```
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
2, (received & used)
  1.1.1.1 (metric 20) from 1.1.1.1 (1.1.1.5)
    Origin IGP, localpref 100, valid, internal, best, group-best, multipath
    Received Path ID 1, Local Path ID 1, version 160385
    Extended community: LB:1000:128
                        (LB non-transitive AS:bytes/sec:1000:16000.0)
    Originator: 1.1.1.5, Cluster list: 1.1.1.1
Path #2: Received by speaker 0
Not advertised to any peer
2, (received & used)
  1.1.1.2 (metric 20) from 1.1.1.2 (1.1.1.5)
    Origin IGP, localpref 100, valid, internal, multipath
    Received Path ID 1, Local Path ID 0, version 0
    Extended community: LB:1000:64
                        (LB non-transitive AS:bytes/sec:1000:8000.0)
    Originator: 1.1.1.5, Cluster list: 1.1.1.2
```

In the example show output, BGP advertises a link bandwidth value of 16,000.0 bytes per second for the link bandwidth weight 128 and 8,000 bytes per second for the link bandwidth weight 64.

BGP DMZ transitive-bandwidth extended community support

BGP DMZ transitive-bandwidth extended community is a feature that allows BGP to process and make routing decisions based on the available transitive-bandwidth extended community using UCMP.

Table 26: Feature History Table

Feature Name	Release Information	Feature Description
BGP DMZ transitive-bandwidth extended community support	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.

Feature Name	Release Information	Feature Description
BGP DMZ transitive-bandwidth extended community support	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can now enable BGP to process incoming DMZ transitive-bandwidth extended community, allowing bandwidth-aware routing decisions using Unequal Cost Multi-Path (UCMP). The feature allows RPL to manually set the DMZ transitive-bandwidth extended community for BGP neighbors. This extended propagation supports multivendor interoperability, optimizes traffic distribution, prevents link over utilization, and balances load across available paths. Previously, BGP supported only the non-transitive extended community. The feature introduces these changes: CLI: The **transitive-bandwidth** type is introduced as an extended community in RPL. YANG Data Models: • `Cisco-IOS-XR-um-route-policy-cfg` • `Cisco-IOS-XR-policy-repository-cfg` (see GitHub, YANG Data Models Navigator)

You can now configure BGP to process incoming DMZ transitive-bandwidth extended communities, enabling bandwidth-aware routing decisions through UCMP. This enhancement allows the router to interpret linked bandwidth values that are attached to incoming BGP routes and use them to distribute traffic proportionally across multiple paths based on available capacity.

In addition, the feature extends RPL support to manually set the DMZ transitive-bandwidth extended community for inbound and outbound BGP updates. This extended propagation supports multivendor interoperability, optimizes traffic distribution, and prevents link over utilization.

Benefits of BGP DMZ transitive-bandwidth extended community

The key benefits of the feature are:

- Ensures multivendor interoperability and utilizes the DMZ bandwidth value across heterogeneous networks.
- Adds a weight that enables RIB or FIB to intelligently load balance traffic across multiple paths.
- Ensures consistent network policies across domains.

How BGP DMZ transitive-bandwidth extended communities work

Summary

The key components involved in the process are:

- BGP router: Assigns and propagates DMZ transitive-bandwidth values.
- BGP peers: Receive and utilize the propagated bandwidth attributes.
- RIB/FIB: Use bandwidth values to make intelligent load balancing decisions.

The BGP DMZ transitive-bandwidth extended community workflow involves assigning bandwidth values, propagating them, and then using these values for traffic load balancing.

Workflow

These stages describe how BGP DMZ transitive-bandwidth extended community works.

1. Policy-based bandwidth assignment: A BGP router assigns a DMZ transitive-bandwidth value to BGP routes using an egress or ingress RPL.
2. Propagation of the bandwidth attribute: The router propagates the transitive-bandwidth extended community along with the route to its BGP peers.
3. Traffic is load balanced based on the bandwidth: Routers use the bandwidth values to balance traffic across multiple paths, favoring higher-capacity links.

Result

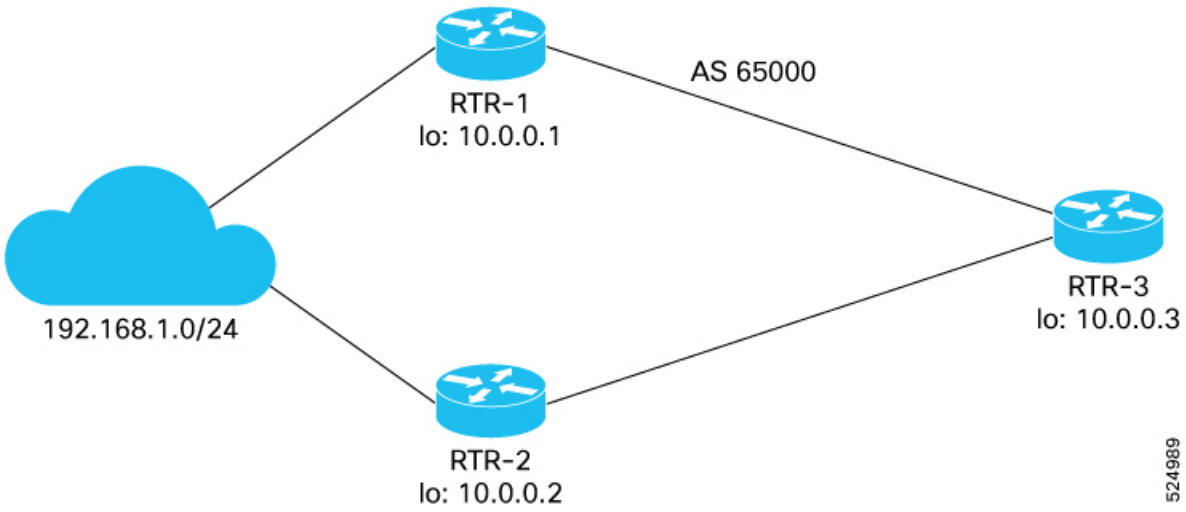
The system achieves optimized traffic distribution and load balancing across available paths based on bandwidth.

Topology for BGP path selection using UCMP

To describe a sample network configuration that demonstrates bandwidth-aware BGP path selection using UCMP.

The figure shows an example topology for bandwidth-aware BGP path selection using UCMP.

Figure 18: Example topology for bandwidth-aware BGP path selection using UCMP



The topology consists of three BGP routers: RTR-1, RTR-2, and RTR-3. All the routers are within the same AS (65000), forming an Internal Border Gateway Protocol network.

- RTR-1 and RTR-2 learn and advertise an external prefix 192.168.1.0/24. The routers apply an RPL outbound toward RTR-3, setting the link-bandwidth value through the transitive-bandwidth extended community.
- RTR-3 receives the prefix from RTR-1 and RTR-2, evaluates their advertised Link Bandwidth (LB) values, and uses UCMP to make routing decisions.

  Consider a scenario where RTR-1 advertises an LB value of `65000:1250000000`, and RTR-2 advertises `65000:2500000000`. Based on these LB values, RTR-3 forwards approximately 33% of the traffic to RTR-1 and 67% to RTR-2, effectively sending twice as much traffic to RTR-2. This traffic distribution allows RTR-3 to balance the load according to the relative bandwidth of each link.

Configure BGP DMZ transitive-bandwidth extended community

Before you begin

Ensure that the network supports DMZ bandwidth extended communities for signaling link bandwidth.

Procedure

Step 1 Configure the transitive-bandwidth extended community for a specific prefix.

- Configure the transitive-bandwidth extended community directly within the route policy.

  In the sample configuration for RTR-1, the route policy for RTR-1 advertises prefix 192.168.1.0/24, and attaches a transitive-bandwidth extended community to that route. The ASN is 65000 and the link-bandwidth is 1,250,000,000 bytes per second.

```
RTR-1#config
RTR-1(config)#route-policy SET-LB
RTR-1(config-rpl)#if destination in (192.168.1.0/24) then
```

```
RTR-1(config-rpl-if)#set extcommunity transitive-bandwidth (65000:1250000000)
RTR-1(config-rpl-if)#endif
RTR-1(config-rpl)#end-policy
```

- Configure the transitive-bandwidth extended community using the **extcommunity-set** command, and create a route policy that applies the transitive-bandwidth extended community to the route.

  In the sample configuration for RTR-2, the extended community set is LB that matches the transitive-bandwidth extended community. The route policy SET-LB applies the transitive-bandwidth extended community that is defined by the named set LB to RTR-2.

```
RTR-2#config
RTR-2(config)#extcommunity-set transitive-bandwidth LB
RTR-2(config-ext)#65000:2500000000
RTR-2(config-ext)#end-set
RTR-2(config)#route-policy SET-LB
RTR-2(config-rpl)#if destination in (192.168.1.0/24) then
RTR-2(config-rpl-if)#set extcommunity transitive-bandwidth LB
RTR-2(config-rpl-if)#endif
RTR-2(config-rpl)#end-policy
```

Step 2 Run the **router bgp** command on RTR-1 and RTR-2 to enable BGP for the required ASN, and apply the configured route policy to outbound updates for the required neighbor.

Example:

In the sample configuration, BGP is enabled for AS 65000 on RTR-1 and RTR-2, and applies the SET-LB route policy to outbound updates to the neighbor RTR-3 (10.0.0.3).

```
Router(config)#router bgp 65000
Router(config-bgp)#neighbor 10.0.0.3
Router(config-bgp-nbr)#remote-as 65000
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy SET-LB out
```

Step 3 Run the **show running-config** command to verify the running configuration for RTR-1 and RTR-2.

Example:

This is a sample output from RTR-1.

```
route-policy SET-LB
  if destination in (192.168.1.0/24) then
    set extcommunity transitive-bandwidth (65000:1250000000)
  endif
end-policy
!
router bgp 65000
 neighbor 10.0.0.3
  remote-as 65000
  address-family ipv4 unicast
   route-policy SET-LB out
  !
 !
!
```

Example:

This is a sample output from RTR-2.

```
extcommunity-set transitive-bandwidth LB
  65000:2500000000
end-set
!
```

```
route-policy SET-LB
  if destination in (192.168.1.0/24) then
    set extcommunity transitive-bandwidth LB
  endif
end-policy
!
router bgp 65000
 neighbor 10.0.0.3
  remote-as 65000
  address-family ipv4 unicast
   route-policy SET-LB out
  !
 !
!
```

Step 4 Run the **show bgp ipv4 unicast advertised neighbor** command to verify the routes that are advertised from RTR-1 to RTR-3, and RTR-2 to RTR-3.

Example:

The sample show output displays the routes that are advertised from RTR-1 to RTR-3. The next-hop and router ID 10.0.0.1 indicates that the route came from RTR-1. The *extended community: LB_TRANS:65000:10000000* value in kbps confirms that RTR-1 supports and advertises the link-bandwidth attribute.

```
RTR-1#show bgp ipv4 unicast advertised neighbor 10.0.0.3
Fri Apr  4 08:09:06.297 UTC
192.168.1.0/24 is advertised to 10.0.0.3
  Path info:
    neighbor: Local           neighbor router id: 10.0.0.1
    valid  redistributed  best
Received Path ID 0, Local Path ID 1, version 5
  Attributes after inbound policy was applied:
    next hop: 0.0.0.0
    MET ORG AS
    origin: incomplete  metric: 0
    aspath:
  Attributes after outbound policy was applied:
    next hop: 10.0.0.1
    MET ORG AS EXTCOMM
    origin: incomplete  metric: 0
    aspath:
    extended community: LB_TRANS:65000:10000000
```

Example:

The sample show output displays the routes that are advertised from RTR-2 to RTR-3. The next-hop and router ID 10.0.0.2 indicates that the route came from RTR-2. The *extended community: LB_TRANS:65000:20000000* value in kbps confirms that RTR-2 supports and advertises the link-bandwidth attribute.

```
RTR-2#show bgp ipv4 unicast advertised neighbor 10.0.0.3
Fri Apr  4 08:12:03.484 UTC
192.168.1.0/24 is advertised to 10.0.0.3
  Path info:
    neighbor: Local           neighbor router id: 10.0.0.2
    valid  redistributed  best
Received Path ID 0, Local Path ID 1, version 5
  Attributes after inbound policy was applied:
    next hop: 0.0.0.0
    MET ORG AS
    origin: incomplete  metric: 0
    aspath:
  Attributes after outbound policy was applied:
    next hop: 10.0.0.2
```

```
    MET ORG AS EXTCOMM
    origin: incomplete  metric: 0
    aspath:
    extended community: LB_TRANS:65000:20000000
```

Step 5 Run the **show bgp ipv4 unicast** command on RTR-3, to verify the multiple paths and extended community attributes for load-balancing bandwidth advertisement.

Example:

In the sample show output from RTR-3, the highlighted values refer to the transitive-bandwidth extended communities, which are used to make routing decisions using UCMP.

```
RTR-3#show bgp ipv4 unicast 192.168.1.0/24
Fri Apr  4 08:03:49.333 UTC
BGP routing table entry for 192.168.1.0/24
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker                 8            8
Last Modified: Apr  4 08:03:43.608 for 00:00:05
Paths: (2 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    10.0.0.1 (metric 10) from 10.0.0.1 (10.0.0.1)
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 8
      Extended community: LB_TRANS:65000:10000000
                          (LB transitive     AS:bytes/sec:65000:1250000000.0)
  Path #2: Received by speaker 0
  Not advertised to any peer
  Local
    10.0.0.2 (metric 10) from 10.0.0.2 (10.0.0.2)
      Origin incomplete, metric 0, localpref 100, valid, internal, multipath
      Received Path ID 0, Local Path ID 0, version 0
      Extended community: LB_TRANS:65000:20000000
                          (LB transitive     AS:bytes/sec:65000:2500000000.0)
```

Step 6 Run the **show route** command on RTR-3, to verify that the routing decision reflects the intended path preferences set by the BGP policies, when using transitive-bandwidth extended communities for UCMP-based load balancing.

Example:

The sample show output from RTR-3 indicates that the router has learned route 192.168.1.0/24 through BGP (AS 65000), and is using BGP multipath to forward traffic across the two next-hops.

RTR-1 (10.0.0.1) has a weight of 10000000, and RTR-2 (10.0.0.2) has a weight of 20000000. The router uses these values to load balance traffic, sending more traffic through RTR-2 because of the higher weight.

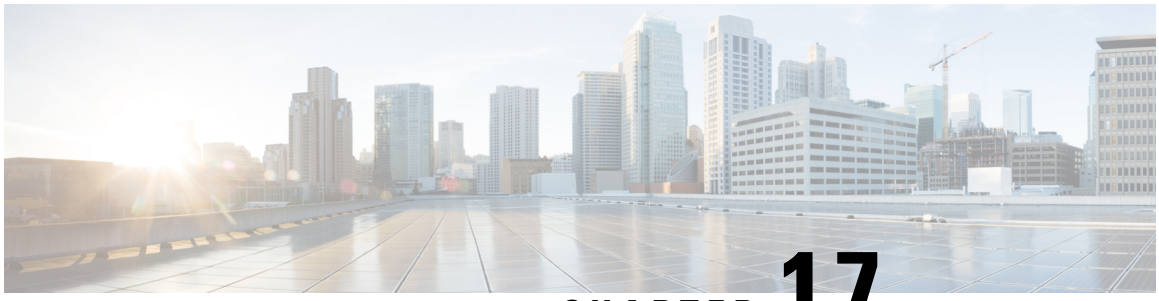
```
RTR-3#show route 192.168.1.0/24
Fri Apr  4 08:03:53.280 UTC

Routing entry for 192.168.1.0/24
  Known via "bgp 65000", distance 200, metric 0, type internal
  Installed Apr  4 08:03:43.605 for 00:00:09
  Routing Descriptor Blocks
    10.0.0.1, from 10.0.0.1, BGP multi path
      Route metric is 0, Wt is 10000000
    10.0.0.2, from 10.0.0.2, BGP multi path
      Route metric is 0, Wt is 20000000
  No advertising protos.
```

Step 7 Run the **show cef** command on RTR-3, to view detailed information about the CEF entry for the IP prefix 192.168.1.0/24.

Example:

In the sample show output, slot 0 is assigned 1 out of 3 buckets, and slot 1 is assigned 2 out of 3 buckets. In other words, the router forwards traffic in a 1:2 ratio, sending approximately 33% through slot 0 and 67% through slot 1.

```
RTR-3#show cef 192.168.1.0/24
Fri Apr  4 08:04:03.111 UTC
192.168.1.0/24, version 20, internal 0x5000001 0x40 (ptr 0x9c449908) [1], 0x0 (0x0), 0x0 (0x0)
 Updated Apr  4 08:03:43.610
 Prefix Len 24, traffic index 0, precedence n/a, priority 4
  gateway array (0x9c2b0798) reference count 1, flags 0x2010, source rib (7), 0 backups
                [1 type 3 flags 0x48441 (0x9c3557d8) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Apr  4 08:03:43.610
 LDI Update time Apr  4 08:03:43.633

    Weight distribution:
    slot 0, weight 10000000, normalized_weight 1
    slot 1, weight 20000000, normalized_weight 2

  Level 1 - Load distribution: 0 1 1
  [0] via 10.0.0.1/32, recursive
  [1] via 10.0.0.2/32, recursive
  [2] via 10.0.0.2/32, recursive

   via 10.0.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 0 NHID 0x0 [0x9c449818 0x0], Internal 0xb329c0a0
    next hop 10.0.0.1/32 via 10.0.0.1/32

    Load distribution: 0 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   FourHundredGigE0/0/0/0    10.10.13.1

   via 10.0.0.2/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 1 NHID 0x0 [0x9c449728 0x0], Internal 0xb329c220
    next hop 10.0.0.2/32 via 10.0.0.2/32

    Load distribution: 0 (refcount 1)

    Hash  OK  Interface                 Address
    1     Y   FourHundredGigE0/0/0/1    10.10.23.2
```



CHAPTER 17

Route Filtering and Policy Enforcement

This chapter describes various methods for implementing routing policies, including BGP route filtering, community and extended community advertisements, and specialized techniques like Remotely Triggered Blackhole (RTBH) filtering. It also covers BGP attribute filtering, error handling, and advanced policy configurations such as conditional matching and VPN route limits.

Routing policy enforcement

Routing policy enforcement is a feature that

- requires configuring inbound and outbound policies for external BGP (eBGP) neighbors
- prevents accidental route acceptance or advertisement, and
- provides an added security measure against configuration omission errors.

Routing policy enforcement behavior

If you do not configure a policy, BGP does not accept any routes from the neighbor, nor does it advertise any routes to it. This enforcement affects only eBGP neighbors, which are neighbors in a different autonomous system than this router. For internal BGP (iBGP) neighbors, which are neighbors in the same autonomous system, BGP accepts or advertises all routes if there is no policy.

Configure routing policy enforcement

Configure BGP routing filtering by applying a route policy.

This task outlines the steps to create and apply a route policy to filter BGP routes, ensuring that only desired routes are accepted or advertised.

Procedure

Step 1 Create a route policy and define its filtering logic.

Example:

```
Router# config
Router(config)# route-policy drop-as-1234
Router(config-rpl)# if as-path passes-through '1234' then
Router(config-rpl-if)# apply check-communities
Router(config-rpl-if)# else
Router(config-rpl-else)# pass
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
```

Step 2 Configures a neighbor IP address as a BGP peer and apply the route policy to inbound or outbound routes for the neighbor.

Example:

```
Router(config)# router bgp 120
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy drop-as-1234 in
Router(config-bgp-nbr-af)# commit
```

BGP routes are filtered according to the defined route policy.

Table policy

Table policy is a feature that

- configures traffic index values on routes as BGP installs them in the global routing table
- supports the BGP policy accounting feature, and
- provides the ability to drop routes from the Routing Information Base (RIB) based on match criteria.

Table policy operational details

You enable this feature using the **table-policy** command. BGP policy accounting uses traffic indices set on BGP routes to track various counters. The ability to drop routes from the RIB is useful in applications such as Remotely Triggered Blackhole (RTBH) filtering to mitigate denial-of-service attacks, or to enforce specific security policies by preventing routes to unwanted destinations from being installed in the local forwarding table.

However, use this feature with caution. It can easily lead to a scenario where BGP advertises routes to neighbors but does not install them in its global routing table and forwarding table. This results in traffic destined for those routes being silently dropped.

Apply policy when updating routing table

Apply a routing policy to routes being installed into the routing table and configure a route policy for BGP neighbors.

This task outlines the steps to configure a policy that influences which routes BGP installs into the global routing table using the **table-policy** command. It also includes an example of how to define a general route policy and apply it to BGP neighbors for inbound and outbound routes.

Procedure

Step 1 Configure the BGP routing process, specifying the autonomous system number and the address family.

Example:

```
Router# configure
Router(config)# router bgp 120.6
Router(config-bgp)# address-family ipv4 unicast
```

Step 2 Apply the routing policy to routes being installed into the routing table.

Example:

```
Router(config-bgp-af)# table-policy tbl-plcy-A
Router(config-bgp)# exit
Router(config)# commit
```

Step 3 Configure a pass-all route policy that accepts and advertises all routes without modifications.

Example:

```
Router(config)# route-policy pass-all
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config)# commit
```

Step 4 Apply the pass-all policy to a BGP neighbor.

Example:

```
Router(config)# router bgp 1
Router(config-bgp)# neighbor 192.168.40.24
Router(config-bgp-nbr)# remote-as 21
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# commit
```

Step 5 Verify the BGP neighbor summary.

Example:

```
Router# show bgp all all summary

Address Family: IPv4 Unicast
============================
```

```
BGP router identifier 10.0.0.1, local AS number 1 BGP generic scan interval 60 secs
BGP main routing table version 41 BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process RecvTblVer bRIB/RIB SendTblVer
Speaker 41 41 41

Neighbor Spk AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down St/PfxRcd
10.0.101.1 0 1 919 925 41 0 0 15:15:08 10
10.0.101.2 0 2 0 0 0 0 0 00:00:00 Idle
```

This command displays eBGP neighbors that do not have both an inbound and outbound policy for every active address family.

The specified policies are applied to routes being installed into the routing table, and to BGP neighbors for inbound and outbound route advertisements.

Policy-based aggregate route management

Policy-based aggregate route management is a feature that

- allows you to configure aggregate routes in the routing table
- marks specific routes as aggregate contributors for a specific destination route via route policy, and
- enables you to set and modify aggregate contributors to a route aggregate address.

Table 27: Feature History Table

Feature Name	Release	Description
Configuring an Aggregate Contributor	Release 7.5.4	You can now configure aggregate routes in the routing table and mark specific routes as aggregate contributors for a specific destination route via route policy. This allows you to set the aggregate contributors to a route aggregate address and modify these routes. You can then use the BGP route policy to tag BGP prefixes before announcing them to the rest of the global network. Earlier, there was no mechanism to identify a more specific route contributing to an aggregate and mark them as aggregate contributors. This feature introduces these changes: • Introduces the **show bgp aggregate-contributors** command • Modifies the **route-policy** command

Aggregate contributor behavior in BGP

Route aggregation in BGP combines several specific routes into one route. You can configure the aggregate routes in the BGP routing table and mark specific routes as aggregate contributors for a specific destination route via route policy.

For example, if you have three prefixes 1.1.1.2/32, 1.1.1.3/32, and 1.1.1.4/32 in the routing table, BGP aggregates them by an aggregate route 1.1.1.0/24 and advertises that route to a peer. Before Cisco IOS XR Release 7.5.4, you could advertise to a peer the more specific route addresses along with the aggregate route (the 1.1.1.X's) or the aggregate route, 1.1.1.0/24.

You can now mark specific routes (the 1.1.1.X's) as aggregate contributors for a specific destination route. This allows you to set the aggregate contributors to a route aggregate address and modify these routes. After setting an aggregate contributor, you also have the option to set BGP attributes (for example, cost community, next-hop, BGP multiple exit discriminator) to the aggregate contributor. This checks the integrity of BGP updates in BGP update messages and optimizes reaction when detecting invalid attributes. You can then apply the inbound policy and the outbound policy to the neighbors.

Restrictions of policy-based aggregate route management

This feature is applicable only for the following Address Family Indicators (AFIs):

- IPv4 unicast
- IPv6 unicast

Configure BGP aggregate contributors

Define a route policy to mark routes as aggregate contributors and associate it with a specific aggregate address route in BGP.

This task guides you through creating a route policy that identifies routes as aggregate contributors and then applying this policy within BGP to an aggregate address. This allows for fine-grained control over how BGP aggregates and advertises routes.

Procedure

Step 1 Define the aggregate contributor route policy.

Example:

```
Router# configure
Router(config)# route-policy aggregate-policy1
Router(config-rpl)# set aggregate-contributor
Router(config-rpl)# end
Router(config-rpl)# commit
```

Step 2 Associate the route policy with an aggregate address in BGP.

Example:

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# aggregate-address 250.2.2.0/24 route-policy aggregate-policy1
Router(config-bgp-af)# commit
```

Step 3 Verify the running configuration.

Example:

```
Router# show running-config
route-policy aggregate-policy1
   set aggregate-contributor
end-policy
!

router bgp 100
  address-family ipv4 unicast
    aggregate-address 250.2.2.0/24 route-policy aggregate-policy1
  !
 !
!
```

Step 4 Verify the BGP routing table entry for a specific prefix to confirm it is an aggregate contributor.

Example:

```
Router#show bgp 250.2.2.1/32

BGP routing table entry for 250.2.2.1/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               247          247
Last Modified: Dec  1 09:00:20.000 for 01:11:55
Paths: (1 available, best #1)
Net is an aggregate-contributor
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192.168.0.5     10.10.10.1
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192.168.0.5     10.10.10.1
  105
    13.0.1.1 from 13.0.1.1 (13.0.1.1)
      Origin IGP, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 1, version 247
      Community: 20:20
      Origin-AS validity: (disabled)
```

Step 5 Display all aggregate contributors of a specific BGP address.

Example:

```
Router#show bgp 250.2.2.0/24 aggregate-contributors

BGP router identifier 192.168.0.2, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 247
BGP main routing table version 247
BGP NSR Initial initsync version 22 (Reached)
BGP NSR/ISSU Sync-Group versions 247/0
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
*> 250.2.2.1/32       13.0.1.1                               0 105 i
*> 250.2.2.2/32       13.0.2.1                               0 105 i
*> 250.2.2.3/32       13.0.3.1                               0 101 i
*> 250.2.2.4/32       13.0.4.1                               0 101 i
*> 250.2.2.5/32       13.0.5.1                               0 102 i
*> 250.2.2.6/32       13.0.6.1                               0 102 i
*> 250.2.2.7/32       13.0.7.1                               0 103 i
*> 250.2.2.8/32       13.0.8.1                               0 103 i
*> 250.2.2.9/32       13.0.9.1                               0 104 i
*> 250.2.2.10/32      13.0.10.1                              0 104 i
*> 250.2.2.11/32      0.0.0.0                  0         32768 ?
*> 250.2.2.12/32      0.0.0.0                  0         32768 ?
*> 250.2.2.13/32      0.0.0.0                  0         32768 ?
```

```
Processed 13 prefixes, 13 paths
```

Set BGP attributes to the aggregate contributor

Setting BGP attributes to the aggregate contributor is an optional step that

- allows you to set or modify BGP attributes, or example, cost community, next-hop, BGP multiple exit discriminator, to the aggregate contributor
- checks the integrity of BGP updates in BGP update messages, and
- optimizes reaction when detecting invalid attributes.

Setting the aggregate contributor to a specific aggregate address route (**aggregate-address address/mask-length route-policy aggregate-route-policy-name**) on a router (Router1) sets the aggregate contributor to the more specific routes on that router (Router1) only. The aggregate contributor then can be used in neighbor In and Out policy to match the BGP prefix with the aggregate contributor on the same router. However, you cannot match the BGP prefix with the aggregate contributor on a remote BGP node (Router2) because the aggregate contributor is not set in the remote node by default.

You can use this optional step to configure BGP attributes for an aggregate contributor via an inbound policy or an outbound policy.

Procedure

Step 1 Configure BGP attributes for an aggregate contributor via an inbound policy in Router1.

Example:

```
Router1# config
Router1(config)# route-policy set_comm_in
Router1(config-rpl)# if aggregate-contributor then
Router1(config-rpl-if)# set community (20:20) additive
Router1(config-rpl-if)# pass
Router1(config-rpl-if)# else
Router1(config-rpl-else)# drop
Router1(config-rpl-else)# endif
Router1(config-rpl)# end-policy
Router1(config)#
Router1(config)# router bgp 100
Router1(config-bgp)# neighbor 13.0.1.1
Router1(config-bgp-nbr)# remote-as 105
Router1(config-bgp-nbr)# address-family ipv4 unicast
Router1(config-bgp-nbr-af)# route-policy set_comm_in in
Router1(config-bgp-nbr-af)# commit
```

Step 2 Configure BGP attributes for an aggregate contributor via an outbound policy in Router1.

Example:

```
Router1# config
Router1(config)# route-policy set_comm_out
Router1(config-rpl)#  if aggregate-contributor then
Router1(config-rpl-if)# set extcommunity rt(200:200) additive
```

```
Router1(config-rpl-if)# pass
Router1(config-rpl-if)# else
Router1(config-rpl-if)# set extcommunity rt(500:500) additive
Router1(config-rpl-else)# pass
Router1(config-rpl-else)# endif
Router1(config-rpl)#end-policy
Router1(config)#
Router1(config)# router bgp 100
Router1(config-bgp)# neighbor 192.168.0.5
Router1(config-bgp-nbr)# remote-as 100
Router1(config-bgp-nbr)# address-family ipv4 unicast
Router1(config-bgp-nbr-af)# route-policy set_extcomm_out out
Router1(config-bgp-nbr-af)# commit
```

Step 3 Verify the running configuration.

- For inbound policy:

```
route-policy set_comm_in
  if aggregate-contributor then
     set community (20:20) additive
     pass
   else
      drop
   endif
 end-policy
 !
router bgp 100
   neighbor 13.0.1.1
   remote-as 105
     address-family ipv4 unicast
        route-policy set_comm_in in
```

- For outbound policy:

```
route-policy set_extcomm_out
  if aggregate-contributor then
     set extcommunity rt (200:200) additive
     pass
  else
    set extcommunity rt (500:500) additive
    pass
  endif
end-policy
!

router bgp 100
  neighbor 192.168.0.5
  remote-as 100
    address-family ipv4 unicast
      route-policy set_extcomm_out out
    !
!
```

Step 4 Verify the BGP routing table entry on Router1 for prefix 250.2.2.1/32 to confirm it is an aggregate contributor and has the expected community attribute.

Example:

```
Router1#show bgp 250.2.2.1/32
Thu Dec  1 10:12:15.374 EST
BGP routing table entry for 250.2.2.1/32
```

```
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                247          247
Last Modified: Dec  1 09:00:20.000 for 01:11:55
Paths: (1 available, best #1)
Net is an aggregate-contributor
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192.168.0.5     10.10.10.1
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192.168.0.5     10.10.10.1
  105
    13.0.1.1 from 13.0.1.1 (13.0.1.1)
      Origin IGP, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 1, version 247
      Community: 20:20
      Origin-AS validity: (disabled)
```

This example shows the BGP attribute (community) being set to aggregate contributor 250.2.2.1/32 via inbound policy. The neighbor inbound policy matches with an aggregate contributor. If it matches, then the route is added with the Community 20:20.

Step 5 Verify the BGP routing table entry on Router2 for prefix 250.2.2.4/32 to confirm it has the expected extended community attribute.

Example:

```
Router2#show bgp ipv4 u 250.2.2.4/32
Thu Dec  8 15:08:40.672 EST
BGP routing table entry for 250.2.2.4/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                174          174
Last Modified: Dec  8 15:06:53.000 for 00:01:47
Paths: (1 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.1
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    12.12.12.4
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.1
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    12.12.12.4
  101, (Received from a RR-client)
    13.0.4.1 (metric 2) from 192.168.0.2 (192.168.0.2)
      OC-RIB Attribute-Index 0
      Origin IGP, localpref 100, valid, internal, best, group-best
      Received Path ID 1, Local Path ID 1, version 174
      Extended community: RT:200:200
```

This example shows the BGP attribute (extended-community) being set to aggregate contributor 250.2.2.4/32 via outbound policy. The neighbor outbound policy matches with an aggregate contributor. If it matches, then the route is added with the Extended community: RT:200:200. This attribute is set before the router (Router2) receives the route.

Remotely triggered blackhole filtering

Remotely triggered blackhole (RTBH) filtering is a technique that

- drops undesirable traffic before it enters a protected network
- quickly drops traffic at the edge of the network, and
- operates based on either source or destination addresses by forwarding traffic to a null0 interface.

These are the types of RTBH filtering:

- Destination-based RTBH filtering: RTBH filtering based on a destination address
- Source-based RTBH filtering: RTBH filtering based on a source address

Benefits of remotely triggered blackhole filtering

This technique enhances network security by

- effectively mitigating Distributed Denial of Service (DDoS) and worm attacks
- quarantining all traffic destined for a target under attack, and
- enforcing blocklist filtering.

How remotely triggered blackhole filtering works

Summary

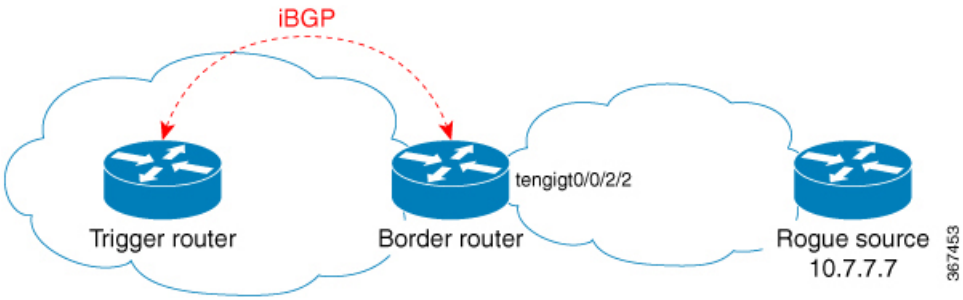
The key components involved in the process are:

- Remotely Triggered Blackhole (RTBH) filtering: A technique to drop undesirable traffic.
- Route Policy Language (RPL): Defines the rules for discarding traffic.
- **set next-hop discard** command: Configures the next-hop of the target prefix to a null interface.
- Routing Information Base (RIB): Stores routing information, updated by the next-hop discard configuration.
- Triggering device: A separate device, typically in a Network Operations Center (NOC), that sends iBGP updates.
- Edge routers: Access and aggregation points that receive iBGP updates from the trigger device.
- Null0 interface: A virtual interface where discarded traffic is sent.

RTBH filtering drops undesirable traffic by rerouting it to a null0 interface, based on route policies and BGP updates from a trigger device.

Workflow

Figure 19: Topology to implement RTBH filtering



Consider this topology, where a rogue router is sending traffic to a border router.

These stages describe how RTBH filtering works.

1. Policy definition: You implement RTBH by defining a route policy (RPL) that uses the set next-hop discard command to discard undesirable traffic at the next-hop.
2. Next-hop configuration: RTBH filtering sets the next-hop of the target prefix to the null interface. Traffic destined for the target is then dropped at the ingress.
3. Inbound policy application: You use the **set next-hop discard** configuration in the neighbor inbound policy. When applied to a path, the RIB updates with the next-hop set to Null0, even if the primary next-hop is unreachable.
4. Best path selection: The RTBH path is considered reachable and becomes a candidate in the best path selection process.
5. Readvertisement: The system readvertises the RTBH path to other peers with either the received next-hop or nexthop-self, based on normal BGP advertisement rules.
6. Deployment scenario: In a typical deployment, an internal Border Gateway Protocol (iBGP) runs at the access and aggregation points. A separate device in the Network Operations Center (NOC) acts as a trigger.
7. Traffic dropping: The triggering device sends iBGP updates to the edge routers, which cause undesirable traffic to be forwarded to a null0 interface and dropped.

Result

The network effectively drops undesirable traffic at the edge, preventing it from entering the protected network.

Configure remotely triggered blackhole filtering

Follow these steps to configure destination-based RTBH filtering:

- Configure the trigger router to initiate RTBH filtering.

  The trigger router defines a static route redistribution policy that sets a community on static routes marked with a special tag. It also configures a static route with this tag for the source prefix to be discarded.

- Configure the border router to apply RTBH filtering based on community matches.

The border router defines a route policy that matches the community set on the trigger router and configures the next-hop to *discard*. This policy is then applied to iBGP peers.

Procedure

Step 1 Configure the trigger router with a static route redistribution policy that sets a community on static routes marked with a special tag, and apply it in BGP.

Example:

```
Router(config)# route-policy RTBH-trigger
Router(config-rpl)# if tag is 777 then
Router(config-rpl-if)# set community (1234:4321, no-export) additive
Router(config-rpl-if)# pass
Router(config-rpl-if)# else
Router(config-rpl-else)# pass
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy

Router(config)# router bgp 65001
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# redistribute static route-policy RTBH-trigger
Router(config-bgp-af)# exit

Router(config-bgp)# neighbor 192.168.102.1
Router(config-bgp-nbr)# remote-as 65001
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy bgp_all in
Router(config-bgp-nbr-af)# route-policy bgp_all out
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config)# commit
```

Step 2 Configure a static route on the trigger router, tagging the source prefix that must be discarded.

Example:

```
Router(config)# router static
Router(config-static)# address-family ipv4 unicast
Router(config-static-afi)# 10.7.7.7/32 Null0 tag 777
Router(config-static-afi)# exit
Router(config-static)# exit
Router(config)# commit
```

Step 3 Configure a route policy in the border router that matches the community set on the trigger router and set the next-hop to *discard*.

Example:

```
Router(config)# route-policy RTBH
Router(config-rpl)# if community matches-any (1234:4321) then
Router(config-rpl-if)# set next-hop discard
Router(config-rpl-if)# else
Router(config-rpl-else)# pass
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
Router(config)# commit
```

Step 4 Apply the route policy on the iBGP peers.

Example:

```
Router(config)# router bgp 65001
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit

Router(config-bgp)# neighbor 192.168.102.2
Router(config-bgp-nbr)# remote-as 65001
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy RTBH in
Router(config-bgp-nbr-af)# route-policy bgp_all out
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config)# commit
```

The border router is configured to discard undesirable traffic based on RTBH communities

BGP community and extended-community advertisements

BGP community and extended-community advertisements are mechanisms that

- specify that community or extended-community attributes should be sent to an eBGP neighbor
- are not sent to an eBGP neighbor by default, and
- are always sent to iBGP neighbors by default.

Guidelines for BGP community and extended-community advertisements

- If you configure the **send-community-ebgp** command for a neighbor group or address family group, all neighbors using that group inherit the configuration.
- Configuring the command specifically for a neighbor overrides the inherited values.
- You cannot configure BGP community and extended-community filtering for iBGP neighbors.
- BGP always sends communities and extended-communities to internal BGP (iBGP) neighbors under VPNv4, MDT, IPv4, and IPv6 address families.

Configure BGP community and extended-community advertisements

Enable the router to send community or extended-community attributes to an eBGP neighbor.

By default, BGP does not send community or extended-community attributes to an eBGP neighbor. This task provides the procedure to enable this advertisement.

Procedure

Step 1 Configure the BGP neighbor, specifying its IP address and remote autonomous system number.

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# address-family ipv6 unicast
```

Step 2 Configure the router to send community attributes or extended community attributes to the specified eBGP neighbor.

- ```
  Router(config-bgp-nbr-af)# send-community-ebgp
  Router(config-bgp-nbr-af)# commit
  ```

- ```
  Router(config-bgp-nbr-af)# send-extended-community-ebgp
  Router(config-bgp-nbr-af)# commit
  ```

BGP attribute filters

BGP attribute filter is a feature that

- checks the integrity of BGP updates in BGP update messages
- optimizes reactions when detecting invalid attributes, and
- filters attributes received in the incoming update message.

Purpose and operation of the attribute filter

The BGP attribute filter feature checks the integrity of BGP updates in BGP update messages. It also optimizes reactions when it detects invalid attributes. BGP update messages contain a list of mandatory and optional attributes. These attributes include MED, LOCAL_PREF, and COMMUNITY, and so on. In some cases, if attributes are malformed, filtering them at the receiving router is necessary. The BGP attribute filter regulates which attributes are accepted in incoming update messages. The attribute filter can also filter any attributes that may potentially cause undesirable behavior on the receiving router.

Some BGP updates are malformed due to incorrect formatting of attributes, such as the Network Layer Reachability Information (NLRI) or other fields in the update message. These malformed updates, when received, cause undesirable behavior on the receiving routers. Such undesirable behavior may occur during update message parsing or during re-advertisement of received NLRIs. In such scenarios, it's better to filter these corrupted attributes at the receiving end.

Configure BGP attribute filtering

To filter malformed BGP attributes received in incoming update messages.

You perform this task to prevent undesirable behavior on receiving routers caused by malformed BGP attributes.

Procedure

Step 1 Enter global configuration mode and specify the autonomous system number to enter BGP configuration mode.

Example:

```
Router# configure
Router(config)# router bgp 100
```

Step 2 Run the **attribute-filter** command with the attribute-filter group name to enter the attribute-filter group configuration mode.

Example:

```
Router(config-bgp)# attribute-filter group ag_discard_med
```

Step 3 Specify a single or a range of attribute codes and an associated action.

Example:

```
Router(config-bgp-attrfg)# attribute 24 discard
```

The allowed actions are:

- Treat-as-withdraw: The system considers the update message for withdrawal. The associated IPv4-unicast or MP_REACH NLRIs, if present, are withdrawn from the neighbor's Adj-RIB-In.
- Discard attribute: The system discards this attribute. The matching attributes alone are discarded, and the rest of the update message is processed normally.

Syslog messages for BGP attribute filtering

A BGP syslog message is a notification that

- indicates an event related to BGP error handling or attribute filtering
- is generated when a router receives a malformed update packet or filters attributes, and
- is rate-limited to prevent excessive logging.

BGP error syslog message

```
%ROUTING-BGP-3-MALFORM_UPDATE : Malformed UPDATE message received from neighbor 192.0.2.1
- message length 90 bytes,
error flags 0x00000840, action taken "TreatAsWithdraw".
Error details: "Error 0x00000800, Field "Attr-missing", Attribute 1 (Flags 0x00, Length 0),
 Data []"
```

BGP attribute filtering syslog message for the discard attribute action

```
4843.46]RP/0/0/CPU0:Aug 21 17:06:17.919 : bgp[1037]: %ROUTING-BGP-5-UPDATE_FILTERED :
One or more attributes were filtered from UPDATE message received from neighbor 192.0.2.1
- message length 173 bytes, action taken "DiscardAttr".
Filtering details: "Attribute 16 (Flags 0xc0): Action "DiscardAttr"". NLRIs: [IPv4 Unicast]
 88.2.0.0/17
```

BGP attribute filtering syslog message for the treat-as-withdraw action

```
[391.01]RP/0/0/CPU0:Aug 20 19:41:29.243 : bgp[1037]: %ROUTING-BGP-5-UPDATE_FILTERED :
One or more attributes were filtered from UPDATE message received from neighbor 192.0.2.1
- message length 166 bytes, action taken "TreatAsWdr".
Filtering details: "Attribute 4 (Flags 0xc0): Action "TreatAsWdr"". NLRIs: [IPv4 Unicast]
```

How malformed BGP updates generate syslog messages

Summary

The key components involved in the process are:

- Router: Receives malformed BGP update packets.
- System: Prints ios_msg to the console and applies rate-limiting.
- ios_msg (ROUTING-BGP-3-MALFORM_UPDATE): The specific syslog message generated for malformed update packets.
- Neighbors: BGP peers from which malformed updates are received.
- Discard Attribute (A5) / Local Repair (A6) actions: Specific responses to certain malformed packets.

The system handles malformed BGP update packets by generating specific syslog messages (ios_msg) that are rate-limited to prevent excessive logging, with different logging behaviors for general malformations versus those leading to specific attribute filtering actions.

Workflow

These stages describe how malformed BGP updates generate syslog messages.

1. A router receives a malformed update packet.
2. The system prints an ios_msg of type ROUTING-BGP-3-MALFORM_UPDATE on the console.
3. The system rate-limits this message to one per minute across all neighbors.
4. If malformed packets trigger Discard Attribute (A5) or Local Repair (A6) actions:
    - The system prints the `ios_msg` only once per neighbor per action.
    - This printing behavior is independent of the number of malformed updates received since the neighbor last reached an Established state.

Result

The system provides controlled and informative logging of BGP malformation events, ensuring you are aware of issues without overwhelming the console with duplicate messages.

BGP update message error management

BGP update message error management is a feature that

- handles error update messages to avoid session reset
- classifies BGP update errors into categories based on factors such as severity, likelihood of update errors, or attribute type, and
- handles errors in each category according to the Internet Engineering Task Force (IETF) Inter-Domain Routing (IDR) draft, *I-D:draft-ietf-idr-error-handling*.

The base BGP specification requires a BGP speaker that receives an update message containing a malformed attribute to reset the session. This action affects both the routes with the malformed attribute and other valid routes exchanged over the session, which is undesirable.

The update message error handling implementation classifies BGP update errors into categories based on factors such as severity, likelihood of update errors, or attribute type.

Errors in each category are handled according to the draft.

The system avoids session reset as much as possible during error handling.

Error handling for some categories is controlled by configuration commands that enable or disable the default behavior.

User-defined Martian address check

Martian addresses are IP addresses that

- are typically dropped by routers to prevent access to certain sites
- include specific IPv4 and IPv6 prefixes, and
- can be configured to be accessible using BGP.

When you configure BGP, you can prevent routers from accessing certain sites with certain specific IP address prefixes. The routers drop packets from such IP addresses, known as Martian addresses.

These IPv4 address prefixes are part of Martian addresses:

- 0.0.0.0/8
- 127.0.0.0/8
- 224.0.0.0/4

These IPv6 address prefixes are part of Martian addresses:

- ::
- ::0002 to ::ffff
- ::ffff:a.b.c.d
- fe80:xxxx
- ffxx:xxxx

By default, routers prevent access to sites with Martian addresses. However, you can enable routers with BGP IPv4 address-family or BGP IPv6 address-family configuration to access these sites by configuring the **default-martian-check disable** command.

Restriction Routers with OSPF or IS-IS protocols cannot access these sites even by having the **default-martian-check disable** command configured.

Disable Martian address check

To allow routers with BGP IPv4 address-family or BGP IPv6 address-family configuration to access sites with Martian addresses.

By default, routers prevent access to sites with Martian addresses. This task configures the router to disable this check.

Before you begin

Follow these steps to allow routes from Martian addresses.

Procedure

Step 1 Enter BGP IPv4 or BGP IPv6 address-family configuration mode.

Example:

```
Router# configure
Router(config)# router bgp 100
```

Step 2 Configure the address-family modifier as unicast.

Example:

```
Router(config-bgp)# address-family ipv4 unicast
```

Step 3 Disable the Martian address check.

Example:

```
Router(config-bgp-af)# default-martian-check disable
Router(config-bgp-af)# commit
```

Step 4 Use the **show bgp ipv4 unicast** command or **show bgp ipv6 unicast** command to verify Martian address check status.

Example:

```
Router# show bgp ipv6 unicast
BGP router identifier 2.2.2.1, local AS number 1 BGP generic scan interval 60 secs
Non-stop routing is enabled BGP table state: Active
Table ID: 0xe0800000 RD version: 29 BGP main routing table version 29
BGP NSR Initial initsync version 4 (Reached) BGP NSR/ISSU Sync-Group versions 0/0 Dampening enabled
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
*>i::/0 1:1:1:1:1:1:1:1 100 0 i
```

```
* i192:1::/112 1.1.1.1 0 100 0 ?
*>i 1:1:1:1:1:1:1:1 0 100 0 ?
* iff11:1123::/64 1.1.1.1 2 100 0 ?
*>i 1:1:1:1:1:1:1:1 2 100 0 ?
```

BGP private AS number management

BGP private AS number management is a feature that

- allows routers belonging to a private Autonomous System (AS) to access the global Internet
- removes private Autonomous System Numbers (ASNs) from the AS path in outgoing update messages, and
- optionally replaces those numbers with the local router's ASN to maintain AS path length.

Private AS numbers are ASNs that

- are used by Internet Service Providers (ISPs) and customer networks to conserve globally unique ASNs
- cannot access the global Internet because they are not unique, and
- range from 64,512 to 65,535.

Private AS number removal and replacement in BGP

Public ASNs are assigned by InterNIC, range from 1 to 64,511, and are globally unique. Private ASNs range from 64,512 to 65,535, are not unique, and cannot appear in the global BGP routing table.

Because BGP best path calculations require unique ASNs, private ASNs must be removed from the AS path before propagating routes to external peers. With this featurem you can configure routers to strip private ASNs from outgoing updates in external BGP (eBGP) sessions. Optionally, the router can replace them with its own ASN to preserve the AS path length.

Private AS number removal and replacement methods

The ability to remove and replace private ASNs from the AS path is implemented through these methods:

- The **remove-private-as** command removes private ASNs from the AS path even if the path contains both public and private ASNs.
- The **remove-private-as** command removes private ASNs even if the AS path contains only private ASNs. This command applies only to eBGP peers. The local router's ASN is then appended to the AS path, which prevents a 0-length AS path.
- The **remove-private-as** command removes private ASNs even if they appear before the confederation segments in the AS path.
- The **replace-as** command replaces the private ASNs removed from the path with the local ASN, retaining the same AS path length.

The feature is configurable per address family for each neighbor. This allows you to apply it to a neighbor in one address family without affecting others. This feature only affects outbound update messages for the configured address family.

Verify private AS number removal or replacement

To confirm that private AS numbers are successfully removed or replaced from the AS path.

After configuring the removal or replacement of private AS numbers, you should verify the changes to ensure proper BGP operation.

Before you begin

Ensure the **remove-private-as** or **replace-as** command has been configured

Procedure

Run these commands to verify that the private AS numbers are removed or replaced.

- Run **show bgp neighbors** command.
- Run **show bgp update-group** command.

The command outputs display information that indicates whether private AS numbers were removed or replaced as intended.

Conditional matching on transitive and non-transitive bandwidth extended communities in BGP RPL

Conditional matching on transitive and non-transitive bandwidth extended communities in RPL is a BGP feature that allows route policies to evaluate and act on specific extended community bandwidth values.

Table 28: Feature History Table

Feature Name	Release Information	Feature Description
Conditional matching on transitive and non-transitive bandwidth extended communities in BGP RPL	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) The feature introduces support for conditional matching on transitive and non-transitive bandwidth extended communities in RPL. You can use this capability to create precise BGP routing policies that evaluate and act on specific bandwidth values, enabling effective and intelligent traffic engineering. The feature introduces these changes: CLI: Conditional matching support on extended communities is introduced in RPL. YANG Data Models: • `Cisco-IOS-XR-um-route-policy-cfg` • `Cisco-IOS-XR-policy-repository-cfg` (see GitHub, YANG Data Models Navigator)

RPL now supports conditional matching on extended communities, including transitive and non-transitive bandwidth types, enabling more granular BGP route control. With this capability, you can define precise BGP routing policies that respond to bandwidth attributes, ensuring effective and intelligent traffic engineering. The feature is useful in multipath, multivendor, and large-scale environments.

For information about transitive-bandwidth support and topology for BGP path selection using UCMP, see BGP DMZ transitive-bandwidth extended community support and Topology for BGP path selection using UCMP.

Configure conditional matching on transitive and non-transitive bandwidths

To configure BGP route policies that conditionally match and act on specific bandwidth extended community values for intelligent traffic engineering.

This task enables granular BGP route control by defining policies that respond to bandwidth attributes. It is useful in environments requiring precise traffic engineering based on network bandwidth, particularly in multipath, multivendor, and large-scale deployments.

Procedure

Step 1 Run the **extcommunity-set** command to configure the transitive-bandwidth extended community definition.

Example:

```
Router#config
Router(config)#extcommunity-set transitive-bandwidth LB10G
Router(config-ext)#65000:1250000000
Router(config-ext)#end-set
```

Step 2 Run the **route-policy** command to create a policy that conditionally matches BGP routes based on the transitive-bandwidth extended community, for use in either inbound or outbound direction.

Example:

The policy checks if a route includes a transitive-bandwidth extended community that matches any entry in the predefined set **LB10G**. If it matches, the policy sets the standard BGP community **65000:10** on the route. You can use this community for route filtering, and policy-based forwarding decisions.

```
Router(config)#route-policy MATCH_LB_10G
Router(config-rpl)#if extcommunity transitive-bandwidth matches-any LB10G then
Router(config-rpl-if)#set community (65000:10)
Router(config-rpl-if)#endif
Router(config-rpl)#done
Router(config-rpl)#end-policy
```

Step 3 Run the **show running-config** command to verify the running configuration.

Example:

```
extcommunity-set transitive-bandwidth LB10G
  65000:1250000000
end-set
!
route-policy MATCH_LB_10G
  if extcommunity transitive-bandwidth matches-any LB10G then
    set community (65000:10)
  endif
  done
end-policy
!
```

Step 4 Run the **router bgp** command to enable BGP for the ASN, and apply the configured route policy to inbound IPv4 unicast updates from the specified neighbor.

Example:

In the example configuration, BGP is set up for ASN 65000. A route policy MATCH_LB_10G is applied to inbound BGP updates received from neighbors 10.0.0.0/24.

```
Router(config)#router bgp 65000
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#maximum-paths ibgp8
Router(config-bgp)#neighbor 10.0.0.0/24
Router(config-bgp-nbr)#remote-as 65000
```

```
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#route-policy MATCH_LB_10G in
```

Step 5 Run the **show running-config** command to verify the running configuration.

Example:

```
router bgp 65000
 address-family ipv4 unicast
  maximum-paths ibgp 8
 !
 neighbor 10.0.0.0/24
  remote-as 65000
  address-family ipv4 unicast
   route-policy MATCH_LB_10G in
  !
 !
!
```

Step 6 Run the **show bgp ipv4 unicast** command to verify the multiple paths and extended community attributes for load-balancing bandwidth advertisement.

Example:

In the sample output, Path 1 includes community value **65000:10**, which indicates that the route-policy matched and applied the configured BGP community. Path 2 does not include this community, indicating that the route-policy did not match the condition.

```
Router#show bgp ipv4 unicast 192.168.1.0/24
Fri Apr  4 08:49:24.609 UTC
BGP routing table entry for 192.168.1.0/24
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker                11           11
Last Modified: Apr  4 08:49:13.608 for 00:00:11
Paths: (2 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    10.0.0.1 (metric 10) from 10.0.0.1 (10.0.0.1)
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 11
      Community: 65000:10
      Extended community: LB_TRANS:65000:10000000
                          (LB transitive     AS:bytes/sec:65000:1250000000.0)
  Path #2: Received by speaker 0
  Not advertised to any peer
  Local
    10.0.0.2 (metric 10) from 10.0.0.2 (10.0.0.2)
      Origin incomplete, metric 0, localpref 100, valid, internal, multipath
      Received Path ID 0, Local Path ID 0, version 0
      Extended community: LB_TRANS:65000:20000000
                          (LB transitive     AS:bytes/sec:65000:2500000000.0)
```

VPN route limit on route reflectors

The VPN route limit is a feature that

- allows you to configure Route Reflectors (RRs) to retain only a certain number of unique network entries for each VPN
- defines this limit by a set of Route Targets (RTs) associated with the VPN, and
- ensures the resources of the RR are used efficiently.

Per-VPN configuration

You can set the maximum number of routes an RR accepts from a particular VPN. This ensures the resources of the RR are used efficiently. This limit can be configured for each VPN. Each VPN can have a unique limit based on its individual requirements.

Selective route dropping

When the number of routes configured for a VPN reaches the limit, the RR drops all later routes learned from that VPN. This drop action is specific to the VPN that has exceeded its limit, and it does not affect other VPNs or active BGP sessions.

How the route count mechanism works

Summary

The key components involved in the process are:

- BGP: Maintains a route count for each unique set of RTs.
- Route Reflector (RR): Accepts or drops paths based on the route count and limit.
- Route Target (RT)-set: A unique set of RTs associated with a VPN.
- Prefixes: Network entries with at least one path tagged with a corresponding RT-set.
- Inbound Route Policy Language (RPL) policy: Evaluates incoming paths and sets the RT-set limit.

BGP maintains a route count for each unique set of Route Targets (RTs) to determine route acceptance or dropping based on a configured VPN route limit.

Workflow

These stages describe how the route count mechanism works.

1. BGP maintains a route count for each unique set of RTs. This count reflects the number of prefixes that have at least one path tagged with the corresponding RT-set.
2. The count increments by one for each prefix. This occurs regardless of the number of paths sharing the same RT-set.
3. When BGP receives a path from a neighbor, it evaluates the path against the inbound RPL policy.
4. The inbound RPL sets an RT-set limit for the path.
5. BGP checks the current count for the RT-set.
6. If the count is below the limit, or if the prefix already has a path with the same RT-set, BGP accepts the path.

7. Otherwise, BGP drops the path.
8. If the number of routes configured for a VPN reaches the limit, the RR drops all subsequent routes learned from that VPN. This drop action is specific to the VPN that exceeded its limit. It does not affect other VPNs or active BGP sessions.
9. In scenarios with multiple RRs, a path accepted by one RR results in identical paths from other RRs also being accepted. This promotes network consistency.

Result

BGP consistently manages VPN routes on RRs, ensuring efficient resource use and adherence to configured limits.

Guidelines and limitations of VPN route limit

Guidelines for VPN route limit configuration

Recommendation: Configure the VPN route limit to be twenty percent higher than the expected scale, for example, 1000 routes instead of 833. This protects the Route Reflector (RR) and Provider Edge (PE) devices.

Note When the VPN route limit is reached, the routes from a neighbor may vary if the neighbor experiences a flap. This happens because route dropping depends entirely on the order in which routes are received.

Limitations of VPN route limit

- When the VPN route limit feature is enabled, active and standby RRs may have different prefixes and paths. This happens because active and standby RRs receive updates independently. The RRs do not guarantee the sequence of prefixes. Therefore, Non-Stop Routing (NSR) is not supported with the VPN Route Limit feature.
- If you modify the policy to reduce the VPN route limit, for example, from 200 to 50 routes, the system enforces the updated limit exclusively on single path networks. Networks with multiple paths are not subject to this new route limit. All existing paths are maintained regardless of the reduced threshold.
- For the same RT-set, if the route limit is not the same due to differing route policies for different neighbors, the routing behavior is nondeterministic.

Configure VPN route limit

To configure a VPN route limit on RRs to control the number of unique network entries for each VPN.

This task helps manage RR resources efficiently by preventing an excessive number of routes from a particular VPN.

Procedure

Step 1 Enable BGP routing.

Example:

```
Router(config)# router bgp 100
```

Step 2 Configure a route policy.

Example:

```
Router(config-bgp)# neighbor 10.1.1.1
Router(config-bgp-nbr)# use neighbor-group RRC
Router(config-bgp-nbr)# address-family vpnv4 unicast
Router(config-bgp-nbr-af)# route-policy vpn-route-limit-policy
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# address-family vpnv6 unicast
Router(config-bgp-nbr-af)# route-policy vpn-route-limit-policy
```

Step 3 Run the **set rt-set route-limit** *limit-value* command in route-policy configuration mode to configure the VPN route limit.

Example:

```
Router# config
Router(config)# route-policy vpn-route-limit-policy
Router(config-rpl)# if extcommunity rt matches-any (111:1) then
Router(config-rpl-if)# set rt-set route-limit 5
Router(config-rpl-if)# else
Router(config-rpl-else)# set rt-set route-limit 6
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
```

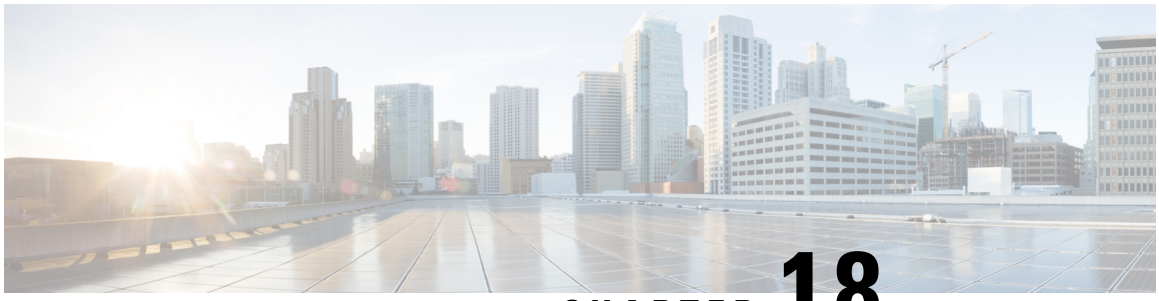
Step 4 Run the s**how bgp vpnv4 unicast rt-set** or **show bgp vpnv4 unicast path rt-set** command to verify the configuration.

- Router# **show bgp vpnv4 unicast rt-set**

```
BGP router identifier 10.3.3.3, local AS number 100
BGP generic scan interval 300 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 4
BGP main routing table version 4
BGP NSR Initial initsync version 3 (Reached)
BGP scan interval 60 secs

Identifier   Route Count   RT-Set
 1           10            111:1
```

- Router# **show bgp vpnv4 unicast path-rt-set**

```
BGP router identifier 10.3.3.3, local AS number 100
BGP generic scan interval 300 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 1269777764
BGP main routing table version 124757
BGP NSR Initial initsync version 3 (Reached)
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        RT-set ID       Route Count
Route Distinguisher: 100:1
*>i51.0.90.0/24       1.1.1.1         1               10
*>i51.0.91.0/24       1.1.1.1         1               10
```



CHAPTER 18

Route Dampening and ECMP Stability Mechanisms

This chapter explains mechanisms that enhance routing stability, including route dampening, ECMP stability techniques, BGP next-hop trigger delays, and delayed BGP route advertisements.

Route dampening

Route dampening is a BGP feature that

- reduces the propagation of unstable, flapping routes across internetworks
- assigns penalties to routes when instability is detected to temporarily suppress advertisements, and
- decays penalties over time so stable routes are reintroduced based on reuse limits.

How route dampening works

Summary

Key components involved in the process are:

- Autonomous systems: AS-1, AS-2, and AS-3.
- eBGP neighbor relationships: Between AS-1 and AS-2, and between AS-2 and AS-3.
- Route to network A: The prefix that flaps and triggers dampening.
- Dampening parameters: Initial penalty (for example, 1,000), suppression limit, reuse limit, half-life, and the history state.

Route dampening limits excessive BGP message propagation caused by route flapping. In a network with three autonomous systems, AS-1, AS-2, and AS-3, dampening penalizes unstable routes and suppresses announcements until stability is restored.

Workflow

These stages describe how route dampening works.

1. Route flap and message flow: The route to network A in AS-1 becomes unavailable. The eBGP neighbor in AS-2 sends a withdraw message, which is propagated to AS-3. When the route reappears, advertisement messages are sent again. Repeated unavailability followed by availability generates many withdraw and advertisement messages.
2. Penalty assignment: When route dampening is enabled, the router assigns an initial penalty to the flapping route (for example, 1,000) and places the route in a history state.
3. Penalty accumulation: Penalties are cumulative. If the cumulative penalty exceeds the suppression limit, the router stops advertising the route to prevent excessive churn.
4. Penalty decay: The penalty value decays with a half-life. When the penalty falls to the reuse limit, the route is re-advertised.
5. State cleanup: When the penalty decays to half of the reuse limit, the router clears the dampening information for that route.

Result

Dampening suppresses repeated announcements and withdrawals for unstable routes, reducing unnecessary BGP message propagation until the route stabilizes.

Note No penalty is applied to a BGP peer reset when route dampening is enabled, even though the reset withdraws the route.

Configure BGP route dampening

Enable and tune route dampening to minimize the impact of flapping routes.

Use address-family configuration to activate dampening with half-life, reuse, suppress, and maximum suppress time values, or attach a route policy.

Before you begin

- Identify the autonomous system number.
- Decide whether to use numeric parameters or a route policy.

Procedure

Step 1 Enter BGP configuration mode and specify the autonomous system number.

Example:

```
Router# configure
Router(config)# router bgp 120
```

Step 2 Configure the address family, IPv4 or IPv6, in unicast mode.

Example:

```
Router(config-bgp)# address-family ipv4 unicast
```

Step 3 Configure dampening parameters or attach a route policy using the **bgp dampening** [*half-life*] [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name* command.

Example:

```
Router(config-bgp-af)# bgp dampening 30 1500 10000 120
Router(config-bgp-af)# commit
```

The router suppresses advertisements for unstable routes and reuses them after penalties decay to the reuse threshold.

BGP next hop trigger delay

BGP next hop trigger delay is a BGP mechanism that

- batches next-hop change notifications to reduce CPU load and avoid unnecessary next-hop walks
- uses the Routing Information Base (RIB) classification of critical and noncritical events, and
- applies a configurable minimum batching interval per address family to control next-hop walk frequency.

How BGP next hop trigger delay works

Summary

The key components involved in the process are:

- Routing Information Base (RIB): Classifies change notifications as critical or noncritical.
- Batching interval: A configured minimum delay that governs next-hop walk scheduling.
- Address families: IPv4 and IPv6 unicast contexts where batching is applied.
- Next-hop walk scheduler: Executes deferred, batched walks based on classification and interval.

BGP next hop trigger delay improves stability by batching next-hop change notifications, reducing CPU load, and controlling how often next-hop walks run per address family.

Workflow

These stages describe how BGP next hop trigger delay works.

1. Classification: The RIB labels each next-hop change notification as critical or noncritical.
2. Interval application: The router applies the configured batching interval to defer next-hop walks for each address family.
3. Batch formation: Deferred notifications accumulate into batches for efficient processing.

4. Interleaved execution: Batched walks are interleaved across address families to prioritize work and avoid contention.
5. Stabilization: Controlled, batched processing reduces churn, improves stability, and supports faster convergence.

Result

Next-hop change notifications are batched and interleaved across address families, lowering CPU utilization and enhancing routing stability and convergence.

Guidelines for BGP next hop trigger delay

Recommendation: Use a nonzero critical next hop trigger delay

- Avoid a critical delay set to 0 in scaled environments or where next-hop changes are frequent.
- A zero delay causes high CPU utilization due to repeated next-hop walks, prevents batching, and increases wait times for address families with nonzero delays, risking traffic blackholing.
- In IPv4, a zero critical delay can slow VPNv4 convergence because IPv4 next-hop updates take precedence.

Effects of zero critical delay

Provide a concise, actionable summary of the operational impacts of setting the BGP next hop trigger delay critical value to 0.

- In scaled deployments or where next-hop changes are frequent, a zero critical delay causes high CPU utilization because each change notification triggers a next-hop walk for address families configured with the **nexthop trigger-delay** *critical 0* command.
- Next-hop change notifications are not batched, which prevents interleaving of next-hop walks in address families with a nonzero delay because those families wait until the zero-delay walks complete.
- Address families with nonzero critical delay values may experience extended wait times before the next-hop walk starts, which can lead to potential traffic blackholing.
- In IPv4, setting the critical delay to 0 can slow VPNv4 convergence because:
  - IPv4 address families are walked as many times as the number of critical alerts raised to BGP.
  - IPv4 next-hop updates for IPv4 prefixes take precedence over VPNv4 prefixes.

Default next hop trigger delay values

Table 29: Default values for next hop trigger delay

	Address families	Value	Notes
Default critical delay	Standard address families	3,000 ms	

	Address families	Value	Notes
Default critical delay	VPNv4, VPNv6	50 ms	Starting in Cisco IOS XR Release 7.10.1, the default critical delay in VPNv4 changed from 0 ms to 50 ms. With this change, all address families have a default nonzero critical delay value.
Default noncritical delay	All address families	10,000 ms	

Use the **show bgp all all nexthops** command to view the critical delay values per address family.

Configure BGP next hop trigger delay

Batch next-hop change notifications to reduce CPU load and avoid unnecessary next-hop walks.

The RIB classifies change notifications as critical and noncritical. A minimum batching interval controls how often next-hop walks run per address family.

Before you begin

Verify the desired delay values for critical and noncritical events per address family.

Procedure

Step 1 Enter the BGP configuration mode and specify the autonomous system number.

Example:

```
Router# configure
Router(config)# router bgp 120
```

Step 2 Specify the address family, IPv4 or IPv6, in unicast mode.

Example:

```
Router(config-bgp)# address-family ipv4 unicast
```

Step 3 Configure **nexthop trigger-delay critical delay** or **nexthop trigger-delay noncritical delay** for batching intervals.

Example:

```
Router(config-bgp-af)# nexthop trigger-delay critical 15000
Router(config-bgp-af)# commit
```

Delay BGP route advertisements

Delay BGP route advertisements is a BGP feature that

- prevents traffic loss by delaying BGP update generation until the Routing Information Base (RIB) is synchronized with the Forwarding Information Base (FIB)
- defers route advertisements during BGP start-up to avoid premature propagation, and
- allows a configurable delay from 1 to 600 seconds.

Table 30: Feature History Table

Feature Name	Release Information	Feature Description
Delay BGP Route Advertisements	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on the Cisco 8712-MOD-M routers.
Delay BGP Route Advertisements	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8212-48FH-M • 8711-32FH-M

Feature Name	Release Information	Feature Description
Delay BGP Route Advertisements	Release 7.5.3	You can now prevent traffic loss due to premature advertising of BGP routes and subsequent packet loss in a network. You can achieve this by setting the delay time of the BGP start-up in the router until the Routing Information Base (RIB) is synchronized with the Forward Information Base (FIB) in the routing table. This delays the BGP update generation and prevents traffic loss in a network. You can configure a minimum delay of 1 second and a maximum delay of 600 seconds. This feature introduces the **update wait-install delay startup** command.

When BGP forwards traffic, it waits for feedback from the Routing Information Base (RIB) until the RIB is ready to forward traffic. After the RIB is ready, BGP sends the route updates to the BGP neighbors and peer-groups. Advertising routes before the RIB is synchronized with the Forwarding Information Base (FIB) can cause traffic loss. To avoid this problem, the router must delay the BGP start-up process to delay the BGP route update generation until the RIB and FIB are synchronized.

To accomplish this, you can configure the **update wait-install delay startup** command to delay BGP update generation. This feature allows you to configure the minimum and maximum delay periods. Use the **show bgp process** command to view the BGP process delay since the last router reload. Set the delay to 1 to 600 seconds to prevent traffic loss.

Restrictions of delay BGP route advertisements

This feature is applicable only for the following Address Family Indicators (AFIs):

- IPv4 unicast
- IPv6 unicast
- VPNv4 unicast
- VPNv6 unicast

Configure BGP route advertisement delay

Delay BGP update generation until the Routing Information Base (RIB) is synchronized, preventing premature route advertisements and traffic loss.

Configure a start-up delay for the desired address family, IPv4, IPv6, VPNv4, or VPNv6, using the BGP address-family submode.

Before you begin

- Determine the BGP autonomous system number.
- Identify the address family to configure.
- Choose the delay value in seconds.

Procedure

Step 1 Specify the BGP autonomous system number and enter BGP configuration mode.

Example:

```
Router# configure
Router(config)# router bgp 1
```

Step 2 Specify the address-family.

Example:

```
Router(config-bgp)# address-family ipv4 unicast
```

Step 3 Schedule the delay of the BGP process to prevent routes from being advertised to peers until RIB is synchronized.

Example:

```
Router(config-bgp-af)# update wait-install delay startup 10
Router(config-bgp-af)# commit
```

Step 4 Verify the running configuration.

Example:

```
Router# show running-config router bgp 1
router bgp 1
 address-family ipv4 unicast
  update wait-install delay startup 10
!
```

Step 5 Run the **show bgp process** command to verify the delay of the BGP process update since the last router reload.

Example:

```
Router# show bgp process

BGP Process Information:
BGP is operating in STANDALONE mode
Autonomous System number format: ASPLAIN
Autonomous System: 100
Router ID: 192.168.0.2 (manually configured)
Default Cluster ID: 192.168.0.2
Active Cluster IDs:  192.168.0.2
------------------------------
------------------------------
Update wait-install enabled:
  ack request 2, ack rcvd 2, slow ack 0
  startup delay 10 secs
```

```
--More—
```

ECMP stability features

Equal-Cost Multi-Path (ECMP) stability features improve forwarding reliability during network reconfigurations, migrations, and path churn. They coordinate BGP and the FIB to avoid out-of-resource conditions, minimize packet loss, and maintain fast convergence. This section includes:

- ECMP out of resource avoidance: Delays best-path selection and hardware programming when resource thresholds are reached to prevent overload.
- ECMP ASN-based prefix download delay: Waits for all ECMP paths from a specified autonomous system (ASN) before inserting prefixes, reducing transient churn and resource spikes.

ECMP out of resource avoidance

ECMP out of resource avoidance is a network resiliency feature that

- tracks hardware resource usage inline in the FIB to provide real-time feedback
- delays BGP best-path selection, route installation into the RIB, and FIB hardware programming when utilization crosses thresholds, and
- uses dampening and Destination-based Load Balancing (DLB) mechanisms to prevent overload and minimize packet loss.

These mechanisms help optimize routing stability and hardware resource usage:

- FIB dampening: A mechanism that consolidates or caches route updates in CPU memory and delays hardware programming when resource usage reaches a configured threshold. FIB dampening is disabled by default and can be enabled through Cisco Express Forwarding (CEF) configuration.
- Dampening switchover: A mechanism that detects when route churn stabilizes and programs stable route updates into hardware. If stability is not detected within the maximum dampening duration, a forced switchover occurs.
- Destination-based Load Balancing (DLB): A protective mode that programs routes with a single forwarding path when hardware resource usage exceeds a configured threshold.

Table 31: Feature History Table

Feature Name	Release Information	Feature Description
ECMP Out of Resource Avoidance	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
ECMP Out of Resource Avoidance	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is supported on: • 8011-4G24Y4H-I
ECMP Out of Resource Avoidance	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8212-48FH-M • 8711-32FH-M

Feature Name	Release Information	Feature Description
ECMP Out of Resource Avoidance	Release 24.2.11	You can now ensure minimum packet loss and service disruption during network reconfigurations or migrations by preventing Equal-Cost Multi-Path (ECMP) Out of Resource (OOR) conditions. This feature allows BGP to delay route updates and FIB to delay programming the routes in hardware when resources are low, thus avoiding system overload. The feature introduces these changes: CLI: • **prefix-ecmp-delay** • **cef load-balancing recursive oor mode dampening-and-dlb** YANG Data Models: • `Cisco-IOS-XR-um-router-bgp-cfg.yang` • `Cisco-IOS-XR-ipv4-bgp-oper.yang` • `Cisco-IOS-XR-fib-common-cfg.yang` • `Cisco-IOS-XR-fib-common-oper.yang` (see GitHub, YANG Data Models Navigator)

The routers can experience transient ECMP resource shortages and traffic drops during data center migrations, maintenance events, or the introduction of new sites that temporarily increase ECMP resource usage. After the network stabilizes, the router recovers from the ECMP spike; however, traffic dropped during an out of resource (OOR) condition does not automatically recover.

How ECMP OOR avoidance works

Summary

The key components involved in the process are:

- FIB inline resource tracking: Measures hardware resource consumption and reports utilization.
- BGP and RIB: Delay best-path selection and route installation when utilization crosses thresholds.
- FIB programming: Defers hardware updates to avoid overload.

- Dampening control: Consolidates updates and manages switchover timings.
- DLB mode: Provides uni-path forwarding under high resource utilization.

ECMP OOR avoidance protects forwarding capacity by monitoring resource usage and temporarily deferring route updates and hardware programming, with optional switchover to DLB when necessary.

Workflow

These stages describe how ECMP OOR avoidance works:

1. Resource monitoring: The FIB tracks hardware utilization and provides real-time feedback.
2. Threshold detection: When utilization reaches the configured threshold, BGP delays best-path selection and route installation into the RIB, and the FIB delays hardware programming.
3. Dampening activation: The FIB consolidates route updates in CPU memory and defers hardware programming to prevent overload.
4. Stability assessment: Dampening switchover checks for churn stability. If stability is detected, the FIB programs the consolidated updates. If stability is not detected within the maximum dampening duration, a forced switchover occurs.
5. DLB engagement: During a forced switchover or when utilization exceeds the DLB threshold, routes are programmed in DLB mode. New route installations may also enter DLB if resource usage is high.
6. Automatic reversion: When hardware resource usage falls below the DLB threshold, affected routes revert to ECMP forwarding.

Result

The router minimizes packet loss and service disruption by deferring route updates and hardware programming under high utilization and by switching to uni-path forwarding when required.

Conditions for DLB programming

Routes are programmed in DLB mode under these conditions:

- New route installation: If hardware resource usage exceeds the configured DLB threshold, program the route in DLB mode to avoid an OOR condition.
- Forced dampening switchover: If hardware resource usage is above the DLB threshold at the end of the maximum dampening duration, program routes in DLB mode.

Additional details:

- DLB operates in a uni-path mode, selecting a single forwarding path to protect against OOR conditions.
- The system automatically switches between DLB and ECMP based on current hardware resource utilization.

Limitations and guidelines of ECMP OOR resource accounting

ECMP OOR resource accounting limitations

- Use ECMP out-of-resource (OOR) accounting primarily in deployments without MPLS in the path. If MPLS is present and the system detects approximately 1,000 or more MPLS link-down indications

(LDIs), the platform increases the resource count to account for maximum MPLS paths only after it observes considerable usage to avoid misclassifying internal labels (for example, BFD internal label) as an MPLS deployment.

- Rely only on FIB recursive and non-recursive LDI accounting. Objects and features that reserve ECMP or members are not included, for example, Layer 2.
- Expect differences between inline FIB resource accounting and SDK resource accounting shown by the **show controller npu resource** command.
- Do not assume FIB transitions LDIs between load-balancing levels, for example, SHLDI to REC_SHLDI to PHLDI. If such a transition occurs, the system disables resource monitoring accounting and issues a warning because counters differ across levels and transitions can create inaccuracies.
- Resource accounting does not apply to management interfaces or special (drop) adjacencies.

Link utilization risks and operational guidelines

- Caution: When DLB mode is active, ECMP path spreading is not available, which can increase the risk of link over-subscription as traffic concentrates on a single path.
- Recommendation: Configure thresholds and dampening durations to balance stability with convergence. The default maximum dampening duration is 5 minutes.

Configure ECMP OOR avoidance in BGP

Configure an ECMP delay duration and a resource usage threshold to prevent out-of-resource (OOR) conditions and reduce packet loss.

The **prefix-ecmp-delay** command is supported only under global AFI/SAFI for IPv4 and IPv6. When the threshold is exceeded, programming of new routes into hardware is deferred for the configured interval.

Before you begin

- Determine the BGP autonomous system number.
- Choose the address family.
- Select the delay interval (milliseconds) and the OOR threshold (percent).

Follow these steps to configure ECMP delay duration and the resource usage threshold limit.

Procedure

Step 1 Specify the autonomous system number and enter BGP configuration mode.

Example:

```
Router# configure
Router(config)# router bgp 100
```

Step 2 Specify the address-family.

Example:

```
Router(config-bgp)# address-family ipv4 unicast
```

Step 3 Run the **prefix-ecmp-delay** *interval_value* **oor-threshold** *threshold_value* command to configure the ECMP delay duration and the OOR threshold value.

Example:

```
Router(config-bgp-af)# prefix-ecmp-delay 10000 oor-threshold 30
```

In this sample configuration, when the resource usage exceeds a threshold of 30%, programming of new routes into the hardware is delayed by 10 seconds (10000 ms).

Currently, this command is supported only in global Address Family Identifier (AFI) and Subsequent Address Family Identifiers (SAFI) for IPv4 and IPv6.

Step 4 Run the **show bgp ipv4 unicast process detail performance-statistics | b OOR** command or **show bgp ipv4 unicast process detail | b OOR** command to verify the configuration.

a) Run the **show bgp ipv4 unicast process detail performance-statistics | b OOR** command to verify the configuration.

Example:

```
Router# show bgp ipv4 unicast process detail performance-statistics | b OOR

OOR queue Info:
 Oldest Queue Num: 0
 Recent Queue Num: 0
 Prefix count HWM: 40000
 Delayed Paths count: 30680000
 Delayed Nets count: 280000
 Processed Nets count: 270000
 Last delayed Q time: May 29 22:30:23.412
 Last processed Q time: May 29 22:31:35.409
 Last OOR recovery time: ---
 Q-num  Q-size   Expiry-Time
  1     0       ---
  2     0       ---
  3     0       ---
  4     0       ---
  5     0       ---
```

b) Run the **show bgp ipv4 unicast process detail | b OOR** command to verify the configuration.

Example:

```
Router# show bgp ipv4 unicast process detail | b OOR
Fri Jun  7 17:38:18.613 UTC
 OOR Flag 0 OOR Threshold 0
 Prefix Download Delay 10000
Dampening is not enabled
```

Step 5 Run the **show bgp** *location* **detail** command to view the details of BGP prefix delays.

Example:

```
Router# show bgp 209.165.201.9/27 detail
BGP routing table entry for 209.165.201.9/27
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker           18490149     18490149
    Flags: 0x00023201+0x28010000+0x00000000 multipath;
Last Modified: Jul 30 19:17:47.643 for 18:43:25
Last Delayed at: Jul 30 19:10:32.643
Paths: (16 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    10.1 0.7 0.8
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    172:23:1:79::2
```

```
  Path #1: Received by speaker 0
  Flags: 0x3000000001078001+0x00, import: 0x020
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    10.1 0.7 0.8
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    172:23:1:79::2
  9001 64313 56001 58505, (received & used)
    209.165.201.2 from 209.165.201.2 (10.1.1.1), if-handle 0x00000000
      Origin IGP, localpref 100, valid, external, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 18490149
      Origin-AS validity: (disabled)
  Path #2: Received by speaker 0
  Flags: 0x3000000001038001+0x00, import: 0x020
  Not advertised to any peer
  9002 64313 56001 58505, (received & used)
    209.165.200.2 from 209.165.200.2 (10.1.1.2), if-handle 0x00000000
      Origin IGP, localpref 100, valid, external, group-best, multipath
      Received Path ID 0, Local Path ID 0, version 0
      Origin-AS validity: (disabled)
  Path #3: Received by speaker 0
  Flags: 0x3000000001038001+0x00, import: 0x020
  Not advertised to any peer
  9003 64313 56001 58505, (received & used)
    209.165.202.2 from 209.165.202.2 (50.1.1.3), if-handle 0x00000000
      Origin IGP, localpref 100, valid, external, group-best, multipath
      Received Path ID 0, Local Path ID 0, version 0
      Origin-AS validity: (disabled)
  Path #4: Received by speaker 0
  Flags: 0x3000000001038001+0x00, import: 0x020
  Not advertised to any peer
  9004 64313 56001 58505, (received & used)
    209.165.200.6 from 209.165.200.6 (10.1.1.4), if-handle 0x00000000
      Origin IGP, localpref 100, valid, external, group-best, multipath
      Received Path ID 0, Local Path ID 0, version 0
      Origin-AS validity: (disabled)
...
```

The sample output indicates that the BGP prefix download to the RIB has been delayed.

ECMP ASN-based prefix download delay

ASN-based prefix download delay is a BGP feature that

- delays downloading prefixes to the Routing Information Base (RIB) and Forwarding Information Base (FIB) based on autonomous system numbers (ASNs) in an Equal-Cost Multi-Path (ECMP) context
- queues new prefixes or paths until the path count per ASN matches the established neighbor count for that ASN, and
- optimizes resource utilization to reduce traffic drops and minimize network disruption during rapid route arrivals.

Table 32: Feature History Table

Feature Name	Release Information	Feature Description
ECMP out of resource avoidance using ASN-based prefix download delay	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can now ensure minimum packet loss and service disruption during network reconfigurations or migrations by preventing ECMP OOR conditions. The feature allows BGP to delay the download of BGP prefixes into the RIB and FIB until the router learns all paths from a specific ASN. This ASN-based delay dynamically optimizes resource utilization, and actively manages ECMP paths in real-time during network changes. Previously, you could apply a fixed delay to all BGP prefixes using the **prefix-ecmp-delay** command. The feature introduces these changes: CLI: • The **ecmp-delay** submode is introduced in the **address-family** command. • **show bgp as-neighbors** YANG Data Models: `Cisco-IOS-XR-um-router-bgp-cfg` (see GitHub, YANG Data Models Navigator)

When a router receives routes from multiple neighbors in the same AS, it delays RIB/FIB insertion until all paths from that AS are learned, helping prevent transient out-of-resource (OOR) conditions caused by hardware limits.

Unlike prefix-ecmp-delay, which applies a fixed delay to all prefixes, ecmp-delay waits for ASN-based ECMP path completion for smarter route selection and resource allocation.

This table explains the key differences between **ecmp-delay** submode and **prefix-ecmp-delay**.

Table 33: Comparison of ecmp-delay and prefix-ecmp-delay in BGP

Category	ecmp-delay	prefix-ecmp-delay
Delay mechanism	Fixed, AS-based, and platform-oor-threshold delay options.	Fixed delay for all prefixes.
Scope	Downloads paths of a prefix only after learning all the ECMP paths from a given ASN, ensuring optimal route installation.	Applies a uniform delay to all prefixes, regardless of ASN or neighbor grouping.
Configuration	Supports per-ASN filtering with AS-based delay configuration.	Applies to all prefixes globally.
Flexibility	Supports different delay types, such as ASN-based delay, and automatically adjusts delays.	Requires manual tuning of the delay interval for all prefixes.
OOR condition handling	Minimizes the probability of causing OOR issues by ensuring all ECMP paths are learnt or ready.	Might still cause OOR issues if the delay is configured incorrectly.
Impact on network convergence	Minimal impact (smart delay).	Can slow network convergence (fixed delay).

How ASN-based prefix download delay works

Summary

The key components involved in the process are:

- ASN-based grouping: Collects all ECMP paths learned from the same AS.
- Delayed queue: Holds new prefixes/paths until ASN path completion.
- RIB/FIB insertion control: Inserts prefixes after completion or after the configured delay (if applicable).

The feature groups paths by ASN and defers RIB/FIB insertion until all ECMP paths from that ASN arrive, reducing churn and transient OOR events.

Workflow

These stages describe how ASN-based prefix download delay works.

1. Wait for ECMP paths: The router waits for all ECMP paths from an ASN before installing routes.
2. Delay on incomplete sets: If the ASN-based ECMP set is incomplete, the router delays RIB/FIB installation for those prefixes to prevent premature route selection.
3. Forceful download after the configured delay: After the configured delay interval, the router forcefully downloads the prefixes, even if all ECMP paths have not arrived.

4. Completion and insertion: When all ECMP paths from the ASN are present, the router downloads the prefix set to the RIB/FIB.

Result

Batched, ASN-aware insertion reduces transient resource spikes, minimizes packet loss, and keeps ECMP routing stable and efficient.

Benefits of ASN-based prefix download delay

The key benefits of the feature are:

- Delaying RIB insertion can eliminate transient OOR conditions with FIB hardware resources.
- The delay runs automatically for BGP prefix downloads into the RIB/FIB, removing the need to tune a universal fixed delay.

Types of delay in ecmp-delay submode

Within the **ecmp-delay** submode, you can configure these delay types:

- Fixed: Delays prefixes by a set time before inserting them into the RIB or FIB.
- Platform-oor-based: Dynamically adjusts the delay based on hardware resource availability.
- AS-based: Waits for all ECMP paths from an ASN before inserting or downloading the path set or nexthop set to RIB.

Limitations and guidelines for ECMP ASN-based prefix delay

Limitations

- Configure the feature only for IPv4 and IPv6 global address families.
- Do not enable the ecmp-delay submode together with prefix-ecmp-delay in BGP.
- Apply the feature only to eBGP-learned ECMP paths; ASN-based prefix grouping is required.

Usage guidelines

- Choose the delay type carefully based on network design and traffic engineering requirements; you cannot apply multiple delay types simultaneously.
- The router downloads prefixes to the RIB after the specified delay (in milliseconds), even if the ECMP set is incomplete because not all paths from the AS have been learned.
- If you provide an AS list, the feature limits operation to the AS numbers in that list; without an AS list, the router applies the feature to all AS numbers learned on the node.

Configure ECMP ASN-based prefix download delay

Configure ASN-based delay for ECMP prefixes so the router inserts routes into the RIB/FIB only after learning all paths from a given autonomous system (ASN), reducing transient out-of-resource (OOR) events.

The ecmp-delay submode operates under global AFI/SAFI for IPv4 and IPv6. You can optionally scope the delay to a specific AS list. When configured, the router defers RIB/FIB insertion by the specified interval (in milliseconds) or until the ASN path set is complete.

Before you begin

- Identify the BGP autonomous system number.
- Choose the address family.
- Decide the delay interval (milliseconds).
- Optionally define the ASNs to include in an AS list.

Follow these steps to configure ASN-based delay:

Procedure

Step 1 In BGP configuration mode, define the address family to install multiple eBGP paths in the RIB and the forwarding table.

Example:

```
Router(config)#router bgp 65536
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#maximum-paths eibgp 1024 selective route-policy mp_rpl
```

Step 2 (Optional) Run the **as-list** command in the BGP configuration mode to define a list of ASNs that must be considered for **ecmp-delay**.

Example:

```
Router(config)#router bgp 65536
Router(config-bgp)#as-list as-list1
Router(config-bgp-as-list)#100
Router(config-bgp-as-list)#300
Router(config-bgp-as-list)#500
Router(config-bgp-as-list)#600
Router(config-bgp-as-list)#commmit
```

Step 3 Run the **ecmp-delay** command to configure delay in the best path calculation for prefixes with ECMP paths based on the neighbor AS.

- Configure delay to download all BGP prefixes with ECMP paths for all ASN numbers learned on the node. In the configuration, the router delays the RIB or FIB installation for BGP prefixes by 10 milliseconds for all AS numbers that are learned on the node.

  ```
  Router(config-bgp-af)#ecmp-delay
  Router(config-bgp-af-ecmpdelay)#as-based delay 10
  ```

- Configure delay to download BGP prefixes with ECMP paths for specific ASN numbers mentioned in the as-list.

  ```
  Router(config-bgp-af)#ecmp-delay
  Router(config-bgp-af-ecmpdelay)#as-based delay 10 as-list as-list1
  ```

Step 4 Run the **show running-config** command to verify the running configuration.

Example:

```
router bgp 65536
 address-family ipv4 unicast
```

```
  maximum-paths eibgp 1024 selective route-policy mp_rpl
  ecmp-delay
   as-based delay 10
  !
 !
!
```

Step 5 Verify the ECMP ASN-based delay for IPv4 unicast routes.

a) Run the **show bgp ipv6 unicast process** command to verify ECMP as-delay configured for IPv4 unicast routes.

Example:

```
Router#show bgp ipv4 unicast process

BGP Process Information:
BGP is operating in STANDALONE mode
Autonomous System number format: ASPLAIN
Autonomous System: 65536
Router ID: 1.1.1.1 (manually configured)
Default Cluster ID: 1.1.1.1
Active Cluster IDs:  1.1.1.1
Fast external fallover enabled
Platform Loadbalance paths max: 1024
Platform RLIMIT max: 8589934592 bytes
Maximum limit for BMP buffer size: 1638 MB
Default value for BMP buffer size: 1228 MB
Current limit for BMP buffer size: 1228 MB
Current utilization of BMP buffer limit: 0 B
Neighbor logging is enabled
Enforce first AS enabled
AS Path multipath-relax is enabled
Use SR-Policy admin/metric of color-extcomm Nexthop during path comparison: disabled
Default local preference: 100
Default keepalive: 30
Graceful restart enabled
Restart time: 1
Stale path timeout time: 0
RIB purge timeout time: 600
Non-stop routing is enabled

ExtComm Color Nexthop validation: RIB
Update delay: 1
Generic scan interval: 60
Configured Segment-routing Local Block: [0, 0]
In use Segment-routing Local Block: [15000, 15999]
Platform support mix of sr-policy and native nexthop: No
  Last insert into reset queue: Mar 17 10:03:29.542, removed at Mar 17 10:03:29.542

Address family: IPv4 Unicast
AS based ECMP Download Delay configured
OOR Flag 0 OOR Threshold 0
Prefix Download Delay 10
Selective EIBGP multipath enabled
Dampening is not enabled
Client reflection is enabled in global config
Dynamic MED is Disabled
Dynamic MED interval : 10 minutes
Dynamic MED Timer : Not Running
Dynamic MED Periodic Timer : Not Running
Scan interval: 60
Total prefixes scanned: 3811
Prefixes scanned per segment: 100000
Number of scan segments: 1
Nexthop resolution minimum prefix-length: 0 (not configured)
```

```
IPv6 Nexthop resolution minimum prefix-length: 0 (not configured)
Main Table Version: 399620
Table version synced to RIB: 399620
Table version acked by RIB: 399620
IGP notification: IGPs notified
RIB has converged: version 84
RIB table prefix-limit reached ?  [No], version 0
RPKI version 3361
RPKI soft-reconfig version 3361
Origin-AS validation is enabled for this address-family
Permanent Network Enabled
Label alloc mode: per-prefix
BGP NSR scoped sync stats:
   Scoped Sync last msg failed: 0
   Scoped Sync last msg resumed: 0
   Scoped Sync default route stopped: 0
   Scoped Sync default route resumed: 0
   Scoped Sync default route lookup failure: 0

OC-RIB Telemetry Neighbor Outbound Attributes Pool summary:
                           Alloc           Free
Pool 25:                   0               0
Pool 49:                   0               0
Pool 73:                   0               0
Pool 97:                   0               0
Pool 121:                  0               0
Pool 145:                  0               0
Pool 169:                  0               0
Pool 193:                  0               0
Pool 217:                  0               0
Pool 241:                  0               0

Number of Paths having particular number of OCRIB out attributes:
                           Paths
1 Out Attrs:               1476400096

Node                Process     Nbrs Estb Rst Upd-Rcvd Upd-Sent Nfn-Rcv Nfn-Snt
node0_RP0_CPU0      Speaker      188  148   2   177737     9562       0       0
```

The sample output is for IPv4 unicast routes configured for ECMP as-delay. In the sample ouput, **Prefix Download Delay 10** indicates that the router delays the RIB or FIB installation for BGP prefixes by 10 milliseconds for all AS numbers that are learned on the node.

b) Run the **show bgp ipv6 unicast process** command to verify ECMP as-delay configured for IPv6 unicast routes.

Example:

```
Router#show bgp ipv6 unicast process
Mon Mar 17 17:23:59.146 UTC

BGP Process Information:
BGP is operating in STANDALONE mode
Autonomous System number format: ASPLAIN
Autonomous System: 65536
Router ID: 1.1.1.1 (manually configured)
Default Cluster ID: 1.1.1.1
Active Cluster IDs:  1.1.1.1
Fast external fallover enabled
Platform Loadbalance paths max: 1024
Platform RLIMIT max: 8589934592 bytes
Maximum limit for BMP buffer size: 1638 MB
Default value for BMP buffer size: 1228 MB
Current limit for BMP buffer size: 1228 MB
Current utilization of BMP buffer limit: 0 B
Neighbor logging is enabled
```

```
Enforce first AS enabled
AS Path multipath-relax is enabled
Use SR-Policy admin/metric of color-extcomm Nexthop during path comparison: disabled
Default local preference: 100
Default keepalive: 30
Graceful restart enabled
Restart time: 1
Stale path timeout time: 0
RIB purge timeout time: 600
Non-stop routing is enabled

ExtComm Color Nexthop validation: RIB
Update delay: 1
Generic scan interval: 60
Configured Segment-routing Local Block: [0, 0]
In use Segment-routing Local Block: [15000, 15999]
Platform support mix of sr-policy and native nexthop: No
  Last insert into reset queue: Mar 17 10:03:29.542, removed at Mar 17 10:03:29.542

Address family: IPv6 Unicast
AS based ECMP Download Delay configured
OOR Flag 0 OOR Threshold 0
Prefix Download Delay 10
Selective EIBGP multipath enabled
Dampening is not enabled
Client reflection is enabled in global config
Dynamic MED is Disabled
Dynamic MED interval : 10 minutes
Dynamic MED Timer : Not Running
Dynamic MED Periodic Timer : Not Running
Scan interval: 60
Total prefixes scanned: 3090
Prefixes scanned per segment: 100000
Number of scan segments: 1
Nexthop resolution minimum prefix-length: 0 (not configured)
IPv6 Nexthop resolution minimum prefix-length: 0 (not configured)
Main Table Version: 407943
Table version synced to RIB: 407943
Table version acked by RIB: 407943
RIB has converged: version 43
RIB table prefix-limit reached ?  [No], version 0
RPKI version 3361
RPKI soft-reconfig version 3361
Origin-AS validation is enabled for this address-family
Permanent Network Enabled
Label alloc mode: per-prefix
BGP NSR scoped sync stats:
   Scoped Sync last msg failed: 0
   Scoped Sync last msg resumed: 0
   Scoped Sync default route stopped: 0
   Scoped Sync default route resumed: 0
   Scoped Sync default route lookup failure: 0

OC-RIB Telemetry Neighbor Outbound Attributes Pool summary:
                           Alloc           Free
Pool 25:                   0               0
Pool 49:                   0               0
Pool 73:                   0               0
Pool 97:                   0               0
Pool 121:                  0               0
Pool 145:                  0               0
Pool 169:                  0               0
Pool 193:                  0               0
Pool 217:                  0               0
```

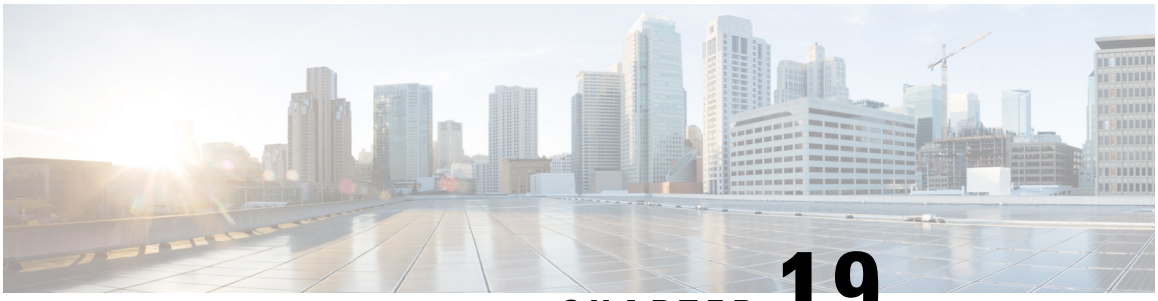
```
Pool 241:                  0              0

Number of Paths having particular number of OCRIB out attributes:
                           Paths
1 Out Attrs:               3371410339

Node                Process     Nbrs Estb Rst Upd-Rcvd Upd-Sent Nfn-Rcv Nfn-Snt
node0_RP0_CPU0      Speaker      188  148   2   177737     9562       0       0
```

c) Run the **show bgp as-neighbors** command to view BGP neighbor relationships grouped by AS.

Example:

```
Router#show bgp as-neighbors
Wed Nov 20 21:10:58.133 UTC
AS: 4294967291, Neighbors: 64, Established: 0
Last updated: Nov 20 00:25:35.285
Neighbor                                        State                   Last state change
31.0.2.2                                         Idle                    Nov 20 19:33:34.119
…
31.0.65.2                                       Idle                    Nov 20 19:33:34.025

AS: 4294967292, Neighbors: 64, Established: 0
Last updated: Nov 20 00:25:35.285
Neighbor                                        State                   Last state change
32.0.2.3                                        Active                  Nov 20 19:33:39.613
…
32.0.65.3                                       Active                  Nov 20 19:33:39.540
```

The router defers RIB/FIB insertion for prefixes with ECMP paths by ASN, either until all paths from the ASN are learned or until the configured delay expires, reducing transient resource spikes and improving stability.



CHAPTER 19

Handling BGP Slow Peers

This chapter covers strategies for managing BGP slow peers, including detection, isolation from update groups, and eBGP session resets in response to link failures.

BGP slow peer management

BGP slow peer management is a Border Gateway Protocol (BGP) mechanism that

- groups neighbors into update groups based on address-family configuration and shared update content
- formats updates once per group, transmits them to all members, and retains updates until all members acknowledge receipt, and
- mitigates the impact of slow peers to prevent group-wide backlogs and maintain update throughput.

Table 34: Feature History Table

Feature Name	Release	Description
BGP Slow Peer	Release 7.9.1	BGP neighbors are grouped together to optimize update generation. BGP peers process the incoming BGP update messages at different rates. A slow peer is a peer that is processing incoming BGP update messages very slowly over a long period of time compared to other peers in the update sub-group. BGP slow peer handling is necessary to reduce the impact of the slow peer on the remaining members of the group. This feature introduces the following commands: • **slow peer (BGP neighbor address-family configuration)** • **slow peer (BGP neighbor address-family configuration)** This feature modifies the following commands: • **show bgp neighbors** to display the slow peer configuration state and slow peer detection or processing information. • **show bgp update out** to display the summary of the neighbor address-family update-group, sub-group, or refresh sub-group information

How update groups work

- BGP neighbors are grouped using common criteria, such as the neighbor address-family configuration and processing of the same update messages, to optimize update generation.
- For each group, messages are formatted once and transmitted to all members; messages are deleted only after all members acknowledge receipt.
- Update generation operates per address family. A peer refers to a neighbor address family, and peers in a sub-group are neighbors for the same address family.

Causes of slow peers

- Processor is busy handling other tasks and cannot process updates on time.
- Peers are connected over slow bandwidth links.
- Temporary network congestion affects timely processing.

Effects of slow peers on update groups

- BGP enforces limits on update messages per process, per address family, and per sub-group. The default sub-group message limit is 32 Mbytes.
- If one or more peers are extremely slow to process messages, those messages remain queued until acknowledged by all peers in the sub-group.
- When the sub-group message limit is reached, all neighbors in the sub-group must wait for the slowest peer to catch up, which slows every member of the sub-group.

Slow peer management reduces these impacts by isolating or otherwise mitigating the effects of slow processing peers.

Slow peer types and states

BGP distinguishes configuration types from runtime states for peers that process updates slowly. Understanding both helps you choose the right mitigation and verify behavior in operation.

Slow peer types

- Static slow peer: Moves the peer to its own update group and requires no additional slow-peer handling.
- Dynamic slow peer: Keeps the peer in the original group; when detected as slow, processing occurs in a refresh sub-group.

Slow peer states

- Static slow peer: The neighbor address family is in the static slow peer state.
- Dynamic detected slow peer: The neighbor address family is detected as slow and is being processed.
- Not slow peer: The neighbor address family is not currently slow.

Table 35: Type-to-state alignment

Type	Typical runtime state	Operational behavior
Static	Static slow peer	Isolated in its own update group; no additional slow-peer handling.
Dynamic	Dynamic detected slow peer or Not slow peer	Remains grouped; when detected slow, updates are handled in a refresh sub-group.
None	Not slow peer	No slow-peer handling.

Guidelines for managing slow peers

Queue and update hygiene

- Ensure queues do not retain stale information; prioritize sending only the latest route state during periods of sustained route churn.
- Monitor sub-group backlogs and message limits to identify slow peers early and take corrective action, for example, apply slow-peer handling features.
- Maintain per–address-family hygiene to prevent a single slow peer from degrading update throughput for the entire sub-group.

Handling permanently slow peers

- Do not rely on dynamic slow peer detection for permanently slow peers.
- Isolate permanently slow peers using static slow peer configuration, which moves them to their own update group; apply the same route policy to all such peers.

How slow peer detection and processing works

Summary

The key components involved in the process are:

- Update groups and sub-groups: Neighbors are grouped by address-family and shared update content; messages are formatted once and deleted only after all members acknowledge receipt.
- Slow peer classification: Static or dynamic, with operational states tracked per neighbor address family.
- Refresh sub-group: A child group created to process a slow peer's updates up to the current table version while the parent sub-group continues with new updates.
- Queues: Main queue and parallel slow peer queue used to control advertisement flow and maintain per-route ordering.
- Detection thresholds and limits: Time threshold (default 300 seconds) and a maximum of 16 refresh sub-groups per address family.

Slow peer handling mitigates backlog and delay in BGP update groups by detecting peers that cannot keep up, isolating their processing in refresh sub-groups, and managing queues to preserve ordering and throughput.

Workflow

These stages describe how slow peer detection and processing works:

1. Group formation and baseline behavior: BGP neighbors are grouped by address-family configuration and shared updates; messages are formatted once and retained until all members acknowledge them. This optimizes update generation but can stall on slow peers.
2. Slow peer detection: A dynamic peer is identified as slow when all of these are true:
    - acknowledgments are missing for some messages

- pending messages are yet to be written to TCP
- the time since the last update exceeds the threshold
- the number of refresh sub-groups for the address family is fewer than 16
- the neighbor is not the only member of the sub-group; other members are already marked slow, and
- there are nets awaiting update generation.

3. Refresh sub-group creation: The system creates a refresh sub-group for the detected slow peer to process updates up to the current table version. The parent sub-group continues sending new updates for both slow and non-slow peers. Each slow peer gets its own refresh sub-group. When processing completes, the refresh sub-group is removed.

   See Refresh sub-group states for more information.

4. Queue management: The router moves all messages queued in the peer's main queue to a parallel slow peer queue and advertises them separately, while new messages continue to flow from the main queue. If the peer is detected slow again, the move-and-advertise cycle repeats. Ordering for updates and withdrawals is maintained per route.

   See Detection and queue management details for more information.

5. State tracking and clearing: The *processing slow peer* state is set to true when handling starts and is cleared only after all slow peer updates are advertised and acknowledged; it does not clear on configuration changes.

6. Recovery: The peer is no longer considered slow when all messages in the slow peer queue are advertised and acknowledged. The slow peer queue is deleted, and the refresh sub-group is removed.

Result

Update generation remains responsive and fair; slow peers are isolated without blocking faster peers, preventing sub-group stalls and preserving per-route ordering during churn.

Configure slow peer handling for BGP neighbors

Enable and verify slow peer handling globally or per neighbor address family to mitigate update backlogs and isolate permanently slow peers.

Slow peer handling supports

- global configuration that affects all neighbors and
- per–neighbor address-family configuration that targets a specific peer.

Dynamic detection can optionally use a threshold (seconds) for classification.

Before you begin

- Identify the BGP autonomous system number.
- Determine whether you need a global setting, a per–neighbor address-family setting, or both.
- If using dynamic detection, choose the detection threshold (seconds).

Follow these steps to configure slow peer handling:

Procedure

Step 1 Configure global slow peer handling.

- Enable dynamic slow peer handling for all BGP neighbor address families.

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#slow-peer dynamic
Router(config-bgp)#commit
```

- Disable slow peer handling for all BGP neighbor address families.

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#slow-peer detection-disable
Router(config-bgp)#commit
```

- Enable dynamic slow peer handling with a detection threshold of 120 seconds.

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#slow-peer dynamic threshold 120
Router(config-bgp)#commit
```

Step 2 Configure slow peer handling for a specific neighbor address family.

- Mark a neighbor as a static slow peer (isolates the neighbor in its own update group).

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#neighbor 50.0.0.1
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#slow-peer static
Router(config-bgp-nbr-af)#commit
```

- Disable dynamic slow peer handling for a neighbor address family.

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#neighbor 50.0.0.1
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#slow-peer dynamic disable
Router(config-bgp-nbr-af)#commit
```

- Enable dynamic slow peer handling for a neighbor address family.

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#neighbor 50.0.0.1
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#slow-peer dynamic
Router(config-bgp-nbr-af)#commit
```

- Enable dynamic slow peer handling with a detection threshold of 120 seconds for a neighbor address family.

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#neighbor 50.0.0.1
Router(config-bgp-nbr)#address-family ipv4 unicast
```

```
Router(config-bgp-nbr-af)#slow-peer dynamic threshold 120
Router(config-bgp-nbr-af)#commit
```

Step 3 Run the **show bgp neighbors <neighbor-address> detail** command to view the effective slow peer configuration for a neighbor address family.

Slow peer handling is enabled globally or per neighbor address family as configured. Use the verification command to confirm the effective state for each neighbor.

Slow peer effective configuration state

This table summarizes the effective neighbor AF slow peer configuration or operational state, considering both the slow peer global configuration and the slow peer neighbor AF configuration.

- For example, if the global configuration is *None* and the neighbor configuration is *Static*, then the effective configuration is *Static*.

Table 36: Effective slow peer configuration state

-		Global configuration		
-		[None]	[Dynamic]	[Detection disable]
Neighbor address-family configuration	[None]	Detection-only	Dynamic	None
	[Static]	Static	Static	Static
	[Dynamic]	Dynamic	Dynamic	Dynamic
	[Dynamic Disable]	Detection-only	None	None

The effective neighbor address-family configuration state can be any of the following entries in this table.

- The **show bgp neighbors <neighbor-address> detail** command displays the neighbor address-family configuration states listed here.

Table 37: Effective neighbor address-family configuration state

AF configuration state	Details
Static	• When the effective neighbor address family configuration is Static, the neighbor address family moves to its own update group, isolating it from other neighbors. • To place all slow peers in a single update group, remove the static slow peer configuration and apply the same outbound route policy to all neighbors.

AF configuration state	Details
Dynamic	• When the effective neighbor address family configuration is Dynamic, BGP enables dynamic slow peer processing for that neighbor address family. • If it is detected as slow, BGP processes the neighbor address family in a dedicated refresh sub-group, isolates it from other neighbors in the sub-group, and displays an IOS message indicating the slow state.
Detection-only	• When the effective neighbor address family configuration is Detection-only, BGP logs slow-peer detection and recovery events but applies no mitigation. • The router displays an IOS message when the neighbor address family becomes slow or recovers.
None	When the effective neighbor address family configuration is None, BGP disables slow peer handling for that neighbor address family.

Examples: Configure slow peer handling with combined global and neighbor settings

- Enable dynamic slow peer globally; mark one neighbor as static.

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#slow-peer dynamic
Router(config-bgp)#neighbor 50.0.0.1
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#slow-peer static
Router(config-bgp-nbr-af)#commit
```

- Enable dynamic slow peer globally; disable it for one neighbor address family.

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#slow-peer dynamic
Router(config-bgp)#neighbor 50.0.0.1
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#slow-peer dynamic disable
Router(config-bgp-nbr-af)#commit
```

- Use different dynamic detection thresholds globally and per neighbor.

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#slow-peer dynamic threshold 600
Router(config-bgp)#neighbor 50.0.0.1
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#slow-peer dynamic threshold 120
Router(config-bgp-nbr-af)#commit
```

IOS messages for slow peer events

The system logs messages when a BGP neighbor is detected as a slow peer and when it recovers. These are the available events and corresponding log messages:

- Slow peer detected: `BGP neighbor 50.0.0.1 of vrf default afi IPv4 Unicast is detected as slow-peer`
- Slow peer recovered: `Slow BGP peer 50.0.0.1 of vrf default afi IPv4 Unicast has recovered`

BGP slow peer automatic isolation from update group

BGP slow peer automatic isolation from update group is a BGP feature that

- detects neighbors in an update group that cannot keep up with sustained update generation over time
- automatically moves detected slow peers into a dedicated slow update group and returns them to the original group upon recovery, and
- prevents group-wide stalls by isolating slow peers, allowing non-slow members to continue processing new updates.

Table 38: Feature History Table

Feature Name	Release Information	Feature Description
BGP Slow Peer Automatic Isolation from Update Group	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on the Cisco 8712-MOD-M routers.
BGP Slow Peer Automatic Isolation from Update Group	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8212-48FH-M • 8711-32FH-M

BGP Slow Peer Automatic Isolation from Update Group	Release 7.3.1	A slow peer cannot keep up with the rate at which the router generates BGP update messages over a period of time, in an update group. This feature automatically detects a slow peer in an update group and moves it to a new update group. The feature is enabled on the router, by default. New commands: • **slow-peer detection enable** • **clear bgp slow-peers** Updated commands: • **slow-peer detection disable**

Slow update groups: structure and defaults

- Short-lived churn: A peer that briefly falls behind during events such as connection resets but quickly recovers is not treated as a slow peer.
- Impact of slow peers: The presence of a slow peer increases the number of formatted updates pending transmission in the original update group.
- Group structure:
  - One slow update group exists for each original update group that contains slow peers.
  - Multiple slow peers in the same original group are isolated in separate sub-groups within that slow update group.
  - Slow peers from different original update groups cannot be combined because outbound policy configurations differ.
- Defaults: The feature is enabled by default, and automatic splitting of update groups is enabled by default.

How automatic isolation works

Summary

The key components involved in the process are:

- Original update group: Formats messages once and transmits to all members; deletes messages only after all members acknowledge receipt.
- Slow update group: Dedicated group created when one or more peers in the original group are detected as slow.
- Slow peer sub-groups: Per-peer sub-groups within the slow update group used to process each slow peer independently.
- Recovery action: Moves peers back to the original update group when they catch up.

Automatic isolation mitigates backlog and contention in update groups by detecting slow peers, processing their updates separately in a dedicated slow update group, and restoring them when they recover.

Workflow

These stages describe how automatic isolation works:

1. Detection: The system detects that a peer in the original update group cannot keep pace, causing formatted messages to accumulate.
2. Isolation: The detected slow peer is moved automatically to a new slow update group; if multiple peers are slow, each is placed in its own slow sub-group.
3. Parallel processing: The parent (original) update group continues sending newly modified nets to non-slow peers, while the slow update group processes backlogged updates for slow peers.
4. Recovery: When a slow peer completes processing and catches up, it is moved back to the original update group.

Result

Non-slow peers advance at their regular pace, while slow peers process updates in isolation. This prevents stalls and reduces the risk of backlog growth across the entire group.

Configure slow peer automatic isolation

Detect, isolate, and manage slow BGP peers by configuring global and neighbor-level controls, and verify slow-peer status and queues.

Automatic isolation moves detected slow peers to a dedicated slow update group and returns them to the original group after recovery. Dynamic detection can be enabled globally or per neighbor address family; static marking isolates a specific peer immediately.

Before you begin

- Identify the BGP autonomous system number.
- Determine the neighbor address and address family (AFI/SAFI).
- Decide whether to disable global detection, mark a peer as static, or enable dynamic detection with the permanent option.

Do one or more of the following to configure and manage slow peers.

Procedure

Step 1 Disable slow peer detection globally.

Example:

```
Router# configure
Router(config)# slow-peer-detection disable
```

Any slow peers that are detected are marked as normal peers and moved back to their original update groups. No more slow peers are detected.

Step 2 Mark a neighbor as a static slow peer at the neighbor address-family level.

Example:

```
Router(config)# router bgp 5
Router(config-bgp)# address-family ipv4
Router(config-bgp-af)# neighbor 172.60.2.3
Router(config-bgp-nbr-af)# slow-peer detection disable split-updategroup static
```

The peer becomes part of the slow update group.

Step 3 Enable dynamic detection with permanent option at the neighbor address-family level.

Example:

```
Router(config)# router bgp 5
Router(config-bgp)# address-family ipv4
Router(config-bgp-af)# neighbor 172.60.2.3
Router(config-bgp-nbr-af)# slow-peer detection enable split-update-group
dynamic permanent
```

The peer is moved to a slow update group when detected slow.

- If only the **split-update-group dynamic** command is configured, a dynamically detected slow peer is moved to an existing slow update group or a new one is created. This behavior is enabled by default.
- If the *permanent* keyword is not configured, the peer returns to the original update group after recovery. If the *permanent* keyword is configured, the peer does not return automatically; use the **clear** command to move it back. Use this option if a peer keeps becoming a slow peer and recovering.

Step 4 Clear dynamically detected slow peers.

- Clear all slow peers for a specific AFI/SAFI.

  ```
  Router# clear bgp slow-peers <afi> <safi>
  ```

- Clear all slow peers for a neighbor across AFI/SAFI.

  ```
  Router# clear bgp slow-peers <neighbor-address>
  ```

- Clear a specific AFI, SAFI, and neighbor combination.

  ```
  Router# clear bgp slow-peers <afi> <safi> <neighbor-address>
  ```

Step 5 View the running configuration.

Example:

```
slow-peer-detection disable
router bgp 5
address-family ipv4
 neighbor 172.60.2.3
 slow-peer detection disable split-update-group static
router bgp 5
address-family ipv4
 neighbor 172.60.2.3
 slow-peer detection enable split-update-group dynamic permanent
```

Step 6 View slow-peers summary for neighbors.

Example:

```
show bgp update out neighbor slow-peers brief
Fri Feb 5 00:12:50.830 UTC

VRF "default", Address-family "IPv4 Unicast"
Main routing table version: 9819220
RIB version: 9819220

Neighbor FG SG SG-R UG Status OutQ OutQ-R Version

19.1.3.1 0.4 0.4 --- 0.2 Normal 4864200 0 7073474
19.1.4.1 0.4 0.4 --- 0.2 Normal 5206200 0 7073474
```

Step 7 Check slow-peers across all address families and neighbors, and compare behavior after time passes.

Example:

```
Router# show bgp all all update out neighbor slow-peers
Fri Sep 13 13:57:48.503 PDT
Address Family: IPv4 Unicast
----------------------------
```

After 5 minutes:

```
Router# show bgp all all update out neighbor slow-peers
Fri Sep 13 14:02:23.097 PDT
Address Family: IPv4 Unicast
----------------------------
VRF "default", Address-family "IPv4 Unicast"
Main routing table version: 3329832
RIB version: 3329832
Neighbor 11.11.11.21
Filter-group 0.3, Refresh filter-group ---
Sub-group 0.2, Refresh sub-group ---
Update-group 0.3
Update OutQ: 20447800 bytes (7680 messages) Refresh
update OutQ: 0 bytes (0 messages) Filter-group pending:
7680 messages
```

Slow peer detection is managed globally or per neighbor address family as configured. Detected slow peers are isolated, and recovery behavior is controlled. Use the verification commands to confirm slow-peers status, queues, and group membership.

eBGP session reset on link failure

eBGP session reset on link failure is a BGP feature that

- automatically resets sessions to directly adjacent external peers when a link goes down (fast external fallover)
- lets you disable and re-enable automatic resets using dedicated configuration commands, and
- supports high session counts by increasing packet rate with LPTS PIFIB hardware policing.

eBGP sessions can flap when a node reaches 3,500 sessions with BGP timers set to 10 and 30. Increasing the packet rate helps support more than 3,500 sessions.

Guidelines for fast external fallover

- Use immediate resets (fast external fallover enabled) when rapid failure detection and cleanup are required for adjacent external peers.
- Disable immediate resets if automatic session resets during transient link events or maintenance windows cause instability.
- When approaching or exceeding 3,500 eBGP sessions with aggressive timers (10 and 30), increase LPTS PIFIB hardware policing rates to maintain stability.

Configure fast external fallover behavior

Control whether eBGP sessions reset automatically when a link fails.

Before you begin

Decide whether immediate resets on link-down events are desired for adjacent external peers.

Procedure

Step 1 Disable automatic resets on link failure.

Example:

```
Router# bgp fast-external-fallover disable
```

Step 2 Re-enable automatic resets on link failure.

Example:

```
Router# no bgp fast-external-fallover disable
```

eBGP session reset behavior matches your operational policy for link-down events.

Increase packet rate for high eBGP session counts

Raise LPTS PIFIB hardware policing rates to support more than 3,500 eBGP sessions.

Before you begin

Identify the target location ID, for example, 0/2/CPU0.

Procedure

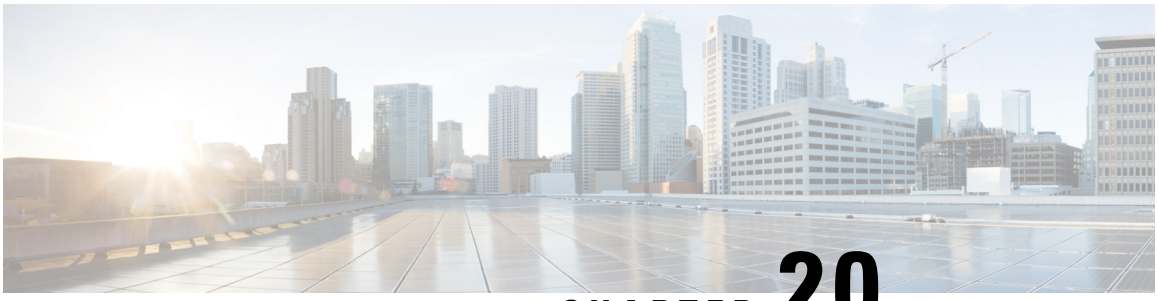
Enter global configuration mode and set BGP flow rates.

Example:

```
Router# configure
Router(config)# lpts pifib hardware police location 0/2/CPU0
Router(config-pifib-policer-per-node)#flow bgp configured rate 4000
```

```
Router(config-pifib-policer-per-node)#flow bgp known rate 4000
Router(config-pifib-policer-per-node)#flow bgp default rate 4000
Router(config-pifib-policer-per-node)#commit
```

The device increases packet handling capacity for BGP flows, helping sustain high eBGP session counts without excessive flapping.



CHAPTER 20

Label Allocation and MPLS Support

This chapter provides comprehensive guidance on BGP labeled unicast (BGP LU) features, enabling scalable MPLS transport across diverse network topologies, including multi-area and multi-AS environments. It details various BGP LU implementations like RSVP-TE, IPv6, MPLS IP POP, and fast convergence with PIC edge. You will also learn how to optimize label allocation and steer BGP control-plane traffic over IP-only paths for enhanced network efficiency.

BGP labeled unicast

The BGP labeled unicast is a routing feature that

- provides MPLS transport between Provider Edge (PE) routers separated by multiple IGP boundaries or autonomous systems
- uses autonomous system border routers (ASBRs) to advertise PE loopback prefixes and their MPLS label bindings via iBGP between area border routers (ABRs) and eBGP between ASBRs, and
- supports multihop eBGP between PEs in different ASes to exchange VPN routes and enables services like 6PE over BGP LU connectivity.

BGP labeled unicast key components

- PE routers are routers at the edge of a provider network that connect to customer networks.
- ASBRs connect different autonomous systems.
- iBGP and eBGP are internal and external Border Gateway Protocol sessions used for route exchange within and between ASes.

Table 39: Feature History Table

Feature Name	Release	Description
BGP Labeled Unicast	Release 24.4.1	Introduced in this release on: Fixed Systems (8700)(select variants only*). This feature enables seamless MPLS transport between Provider Edge (PE) routers across multiple IGP boundaries or autonomous systems by utilizing Autonomous Systems Border Routers (ASBRs) to advertise loopback prefixes and MPLS label bindings through iBGP and eBGP. *This feature is now supported on Cisco 8712-MOD-M Routers.

BGP labeled unicast overview

BGP labeled unicast, also known as unified MPLS, helps you scale MPLS transport across complex network topologies involving multiple IGP areas and AS boundaries. By using BGP LU, you reduce the scale of IGP labeled prefixes and adjacencies, which improves network efficiency.

To avoid unintended scale reduction on routers not configured for BGP LU, you must configure the hw-module command before enabling BGP LU. After configuring this command, you need to restart the router for the changes to take effect.

For example, you can run 6PE and other MPLS VPN services between PEs separated by multiple ASes or IGP areas, simplifying route advertisement and label distribution.

This approach helps you manage large-scale MPLS networks more effectively by lowering the overhead on IGP and BGP processes while maintaining connectivity and service capabilities.

Guidelines for BGP labeled unicast

To ensure proper configuration and compatibility, follow these guidelines and restrictions for the BGP LU feature:

- Use only per-VRF label mode on Cisco 8000; other label modes are not supported.
- Use Label Distribution Protocol (LDP) or Segment Routing (SR) for the transport underlay; do not use Traffic Engineering (TE).
- Do not enable the BGP Prefix Independent Convergence (PIC) edge feature.
- Do not configure L3VPN or 6VPE over BGP LU.
- Enable the BGP Prefix Independent Convergence (PIC) core feature only if needed.
- Do not use the deprecated **label-allocation-mode** command in IOS XR Release 7.4.1 or later; use the **label mode** command under the configured address-family instead.

- Avoid using the deprecated **label-allocation-mode** command to ensure compatibility with current releases.

Features supported by BGP labeled unicast

BGP labeled unicast includes the following supported features:

- BGP LU with inter-AS Option C
- 6PE over MPLS transport using LDP or SR
- BGP PIC core
- L3VPN and L2VPN services over BGP LU using LDP or SR transport

How MPLS connectivity for BGP labeled unicast across multiple OSPF areas works

In large service provider networks, MPLS with BGP labeled unicast enables scalable connectivity across multiple OSPF areas by using label switching and iBGP-based label distribution. This process is essential for connecting remote sites and maintaining efficient, reliable routing in complex environments.

Summary

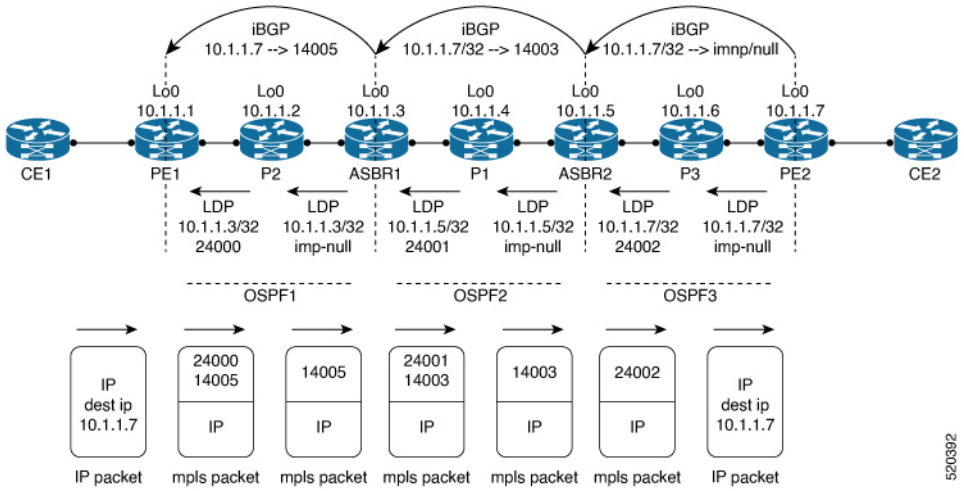
Effective BGP labeled unicast implementation across diverse OSPF areas relies on the coordinated functions of these key components:

- PE1 and PE2: Provider edge routers that establish connectivity across OSPF areas.
- OSPF areas (OSPF1, OSPF2, OSPF3): Separate OSPF instances running in different areas to support routing.
- Label Distribution Protocol (LDP): Provides transport label switching between OSPF areas.
- iBGP: Internal BGP used to advertise labels and loopback addresses between PE2 and PE1 via intermediate routers.
- ASBR1 and ASBR2: Autonomous system border routers that assign and swap labels during packet forwarding.
- P1, P2, P3: Intermediate routers in the MPLS network forwarding iBGP and MPLS traffic.

This coordinated interplay of edge, core, and routing protocols facilitates robust and scalable MPLS transport for BGP labeled unicast across disparate OSPF areas.

Workflow

Figure 20: BGP labeled unicast (Intra-Autonomous System) control plane and data plane



These stages describe how the BGP labeled unicast (intra-autonomous aystem) control plane and data plane works:

1. Establishing iBGP connectivity
   - Actor: PE2, intermediate routers (P3, ASBR2, P1, ASBR1, P2), PE1
   - Action: PE2 sends iBGP updates to PE1 through the path P3 → ASBR2 → P1 → ASBR1 → P2. PE1 learns PE2's loopback address to establish connectivity.
2. Label advertisement and allocation
   - Actor: PE2, ASBR2, ASBR1, PE1
   - Actions:
     - PE2 advertises its loopback address 10.1.1.7 with a BGP label (implicit null) via iBGP to ASBR2.
     - ASBR2 assigns a local label 14003 and advertises it to ASBR1.
     - ASBR1 assigns label 14005 and advertises it to PE1.
     - PE1 learns the prefix and label 14005, with ASBR1 as the BGP next hop.
3. Packet forwarding from PE1 to PE2
   - Actor: PE1, ASBR1, ASBR2
   - Actions:
     - PE1 sends traffic to PE2 with two labels: the BGP-LU label 14005 and the transport LDP label 24000 on top.
     - The transport LDP label carries the packet to ASBR1.
     - ASBR1 swaps the BGP-LU label from 14005 to 14003, applies transport LDP label 24001, and forwards the packet to ASBR2.

- ASBR2 uses an implicit null BGP-LU label and pushes transport label 24002 to forward the packet to PE2.

Result

This process enables PE1 and PE2 to communicate across multiple OSPF areas using MPLS with label switching, ensuring efficient and scalable inter-area connectivity.

How inter-AS connectivity works using eBGP

Summary

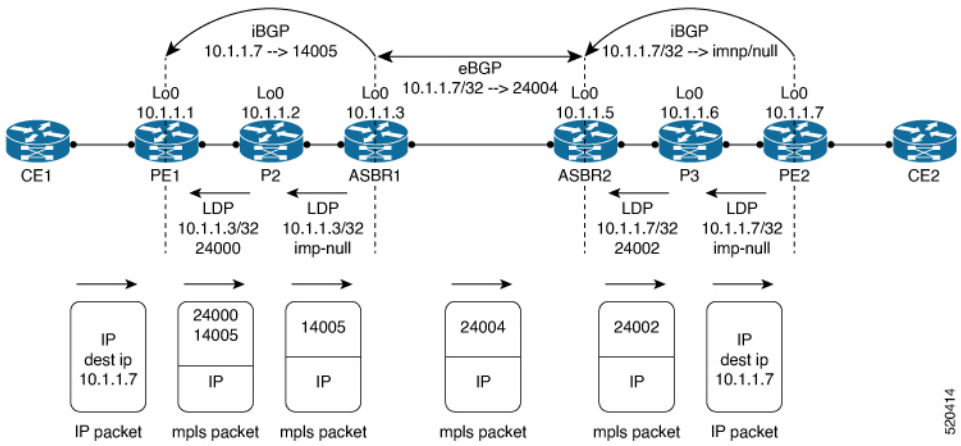
The key components involved in the process are:

- PE1 and PE2: Provider Edge routers serving as the source and destination for IP packets.
- ASBR1 and ASBR2: Autonomous System Border Routers performing eBGP peering, label advertisement, allocation, and swapping.
- LDP (Label Distribution Protocol): Used to signal transport labels across the MPLS path.
- BGP-LU (BGP Labeled Unicast): Facilitates label route advertisement between routers across and within AS boundaries.
- IGP (Interior Gateway Protocol): Supports local MPLS path computation where applicable.

This process enables reliable MPLS connectivity across multiple autonomous systems. eBGP connects ASBRs at AS boundaries for route and label exchange. By combining BGP-LU, LDP, and correct label operations, providers ensure efficient IP packet forwarding between PE routers over a multi-AS MPLS backbone. Understanding label allocation, advertisement, and swapping across ASBRs is key to inter-AS MPLS success.

Workflow

Figure 21: BGP labeled unicast (Intra-Autonomous System Option C) control plane and data plane



These stages describe how inter-AS connectivity using eBGP works:

1. Label advertisement by PE2

- Actor: PE2
- Action: PE2 advertises the BGP-LU label (implicit null) to ASBR2 via iBGP.

2. Label allocation and advertisement by ASBR2
   - Actor: ASBR2
   - Actions:
     - ASBR2 prefers the IGP MPLS path with LDP label 24002.
     - It allocates a local label 24004 for loopback address 10.1.1.7.
     - ASBR2 advertises label 24004 to ASBR1.

3. Label creation and advertisement by ASBR1
   - Actor: ASBR1
   - Actions
     - ASBR1 creates a local label 14005.
     - It advertises label 14005 to PE1

4. Packet forwarding from PE1 to PE2
   - Actor: PE1, ASBR1, ASBR2
   - Actions
     - PE1 sends IP packets with BGP label 14005 and transport label 24000 to ASBR1.
     - ASBR1 swaps the BGP-LU label 14005 to 24004.
     - ASBR2 pushes the LDP label 24002 and forwards the packet to PE2.

Result

This process ensures that IP packets from PE1 are efficiently delivered to PE2 across autonomous system boundaries using eBGP and MPLS label switching, enabling robust and scalable inter-AS connectivity.

How MPLS connectivity with multihop eBGP between multiple ASes works

Summary

The key components involved in the process are:

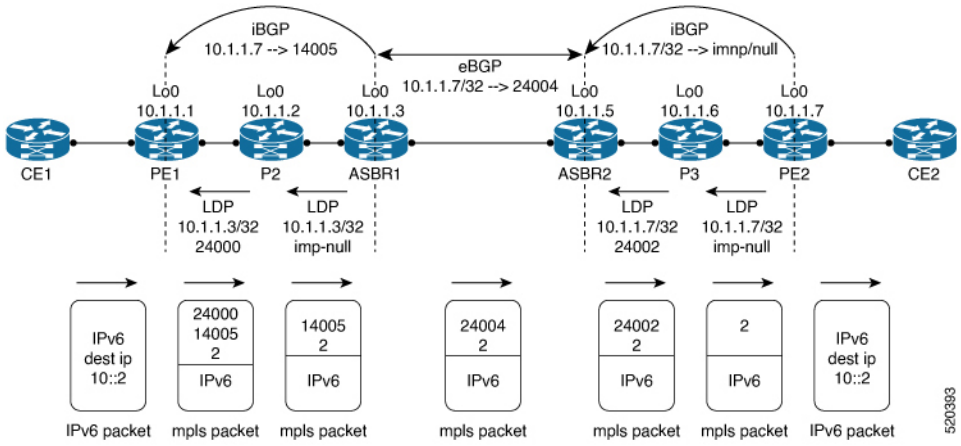
- PE1 router: Initiates label imposition on IPv6 packets and exchanges 6PE routes with PE2 using Multihop eBGP.
- PE2 router: Advertises IPv6 prefixes with explicit null labels and performs IPv6 lookup to forward packets.
- ASBR1 router: Swaps BGP Label Unicast (LU) labels and forwards packets between ASes.

- ASBR2 router: Adds Label Distribution Protocol (LDP) labels and forwards packets toward PE2.
- P3 router: Pops the top LDP label before delivering packets to PE2.

This process describes how PE1 and PE2 use MPLS with multihop eBGP across multiple ASes to enable seamless IPv6 forwarding, focusing on 6PE route exchange and the sequence of label operations.

Workflow

Figure 22: 6PE over BGP LU (inter-AS option C) control plane and data plane



These stages describe how MPLS connectivity with multihop eBGP between multiple ASes works:

1. Establishing a multihop eBGP session and exchanging routes:

   PE1 and PE2 establish a multihop eBGP session across multiple ASes to exchange 6PE routes with associated labels.

2. Advertising IPv6 prefixes by PE2:

   PE2 advertises an IPv6 prefix (e.g., 10::2/128) with the 6PE label set to the IPv6 explicit null label.

3. IPv6 packet arrival at PE1:

   An IPv6 packet destined for 10::2/128 arrives at PE1.

4. Label imposition at PE1:

   The PE1 router imposes labels on the packet in the following order:

   - First, the 6PE label with value 2 (IPv6 explicit null).
   - Next, the BGP label 14005.
   - Finally, the next-hop LDP label 14005 for the BGP LU next hop.

5. Label swapping at ASBR1:

   ASBR1 receives the packet, swaps the BGP-LU label from 14005 to 24004, and forwards it to ASBR2.

6. LDP label imposition by ASBR2:

   ASBR2 adds an LDP label on top of the 6PE label 2 and forwards the packet to P3.

7. Label popping at P3:

   P3 pops the top LDP label so that PE2 receives the packet with only the 6PE explicit null label remaining.

8. IPv6 lookup and forwarding at PE2:

   PE2 performs an IPv6 lookup on the packet and forwards it to the final destination.

Result

This process enables seamless IPv6 packet forwarding between PE routers across multiple ASes by leveraging MPLS with multihop eBGP and label stacking. The correct sequence of label imposition, swapping, and removal ensures that IPv6 traffic is efficiently routed and delivered end to end.

Configure BGP labeled unicast

Enable BGP LU to support labeled IPv6 unicast routing on your router.

BGP labeled unicast extends BGP to distribute labeled routes, allowing routers to forward IPv6 packets using MPLS labels. This capability is essential for scalable and flexible IPv6 routing with MPLS label switching.

Before you begin

- Ensure you have administrative access to the router.
- Confirm the router supports BGP-LU and the required hardware module features.

Follow these steps to configure BGP labeled unicast:

Procedure

Step 1 Enable the BGP-LU hardware profile.

Example:

```
Router(config)# hw-module profile cef bgplu enable
```

Step 2 Restart the router to activate the BGP-LU hardware profile.

Step 3 Enable the BGP-LU feature and configure the BGP router with IPv6 unicast address family settings to support labeled unicast routing.

Example:

```
Router(config)# router bgp 1
Router(config-bgp)# bgp router-id 2001:DB8::1
Router(config-bgp)# address-family ipv6 unicast
Router(config-bgp-af)# redistribute connected route-policy set-lbl-idx
Router(config-bgp-af)# allocate-label all
Router(config-bgp-af)# exit
```

Step 4 Configure a BGP neighbor with an IPv6 address, enabling the labeled-unicast address family and applying inbound and outbound route policies.

Example:

```
Router(config-bgp)# neighbor 2001:DB8::2
Router(config-bgp)# remote-as 1
Router(config-bgp)# update-source Loopback 0
Router(config-bgp)# address-family ipv6 labeled-unicast
```

```
Router(config-bgp)# route-policy pass-all in
Router(config-bgp)# route-policy pass-all out
```

Step 5 Use the **show running-config** to verify the running configuration.

Example:

```
Router# show running-config

hw-module profile cef bgplu enable
!
router bgp 1
 bgp router-id 2001:DB8::1
 address-family ipv6 unicast
 redistribute connected route-policy set-lbl-idx
 allocate-label all
!
 neighbor 2001:DB8::2
 remote-as 1
 update-source Loopback0

!
 address-family ipv6 labeled-unicast
 route-policy pass-all in
 route-policy pass-all out
!
```

BGP labeled unicast is enabled on your router, supporting labeled IPv6 unicast routing with the specified neighbor, and the hardware profile is active.

What to do next

Restart the router after enabling the hardware module profile if you have not already done so to ensure all changes take effect.

BGP labeled unicast over RSVP-TE

A BGP labeled unicast over RSVP-TE is a routing feature that

- enables routers to forward BGP labeled unicast traffic to the BGP-LU next-hop router through RSVP-TE tunnels,
- allows network administrators to select the tunnel path for traffic transport based on your requirements, and
- differentiates traffic by routing packets destined for the tunnel destination address exclusively through Autoroute Announce (AA) tunnels, while routing all other traffic through Forwarding-Adjacency (FA) tunnels.

Table 40: Feature History Table

Feature Name	Release Information	Feature Description

BGP Labeled Unicast over RSVP-TE	Release 24.4.1	Introduced in this release on: Fixed Systems(8200, 8700[ASIC:K100]); Modular Systems (8800 [LC ASIC: P100]) (select variants only*). *This feature is now supported on: • 8212-32FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 8712-MOD-M
BGP Labeled Unicast over RSVP-TE	Release 7.11.1	You can now steer the MPLS traffic as per your requirement instead of relying on what the IGP directs. This feature extends the BGP Labeled Unicast (LU) functionality over RSVP-TE protocol. BGP LU advertises label bindings while RSVP-TE establishes the traffic engineering paths that you specify. This feature allows the provider Edge (PE) routers to forward incoming traffic using the label bindings along the specific path reserved using RSVP-TE. This ability to provide explicit routing ensures optimal use of your network resources. The feature introduces these changes: CLI: • **hw-module profile cef bgplu-over-rsvpte enable** YANG Data Models: • `Cisco-IOS-XR-npu-hw-profile-cfg.yang` (see GitHub, YANG Data Models Navigator)

How BGP labeled unicast path selection and tunnel protection work

BGP-LU with RSVP-TE enables network administrators to select explicit tunnel paths for traffic, assures traffic engineering, and protects service continuity using recovery mechanisms.

Summary

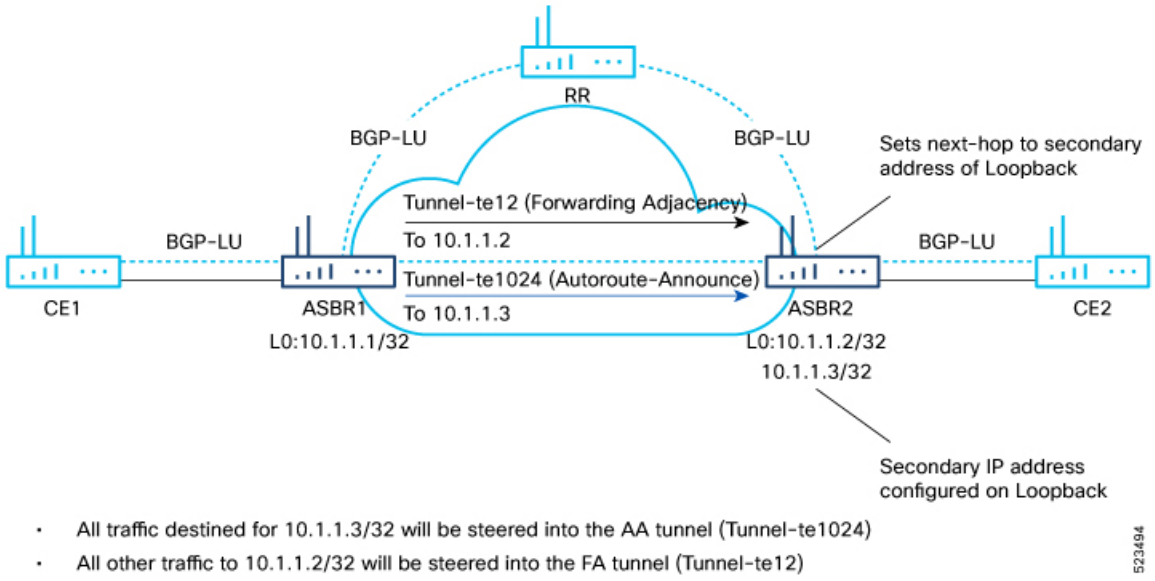
The key components involved in the process are:

- ASBR1: Establishes RSVP-TE tunnels to ASBR2 and forwards traffic to CE2 based on the next-hop address received from ASBR2.
- ASBR2: Receives BGP-LU prefixes and sets next-hop addresses.
- CE1/CE2: Edge routers exchanging traffic.
- RSVP-TE: Tunneling technique for traffic protection.
- Fast Reroute (FRR): Provides resiliency in case of failures.

This process uses BGP-LU and RSVP-TE tunnels to ensure reliable, protected traffic forwarding between customer edge routers across ASBRs, enabling explicit path selection, traffic engineering, and service continuity.

Workflow

Figure 23: BGP labeled unicast over RSVP-TE



These stages describe the process by which BGP-LU and RSVP-TE tunnels are used to establish connectivity, select forwarding paths, and protect traffic between CE1 and CE2 through ASBR1 and ASBR2, ensuring reliable and resilient network service.

1. Establish BGP-LU connections: ASBR1 connects to CE1 and ASBR2 connects to CE2 using BGP labeled unicast.
2. Configure RSVP-TE tunnels: ASBR1 is configured with FA and AA tunnels to ASBR2's primary and secondary IP addresses, respectively.
3. Set BGP-LU next-hop: ASBR2 sets BGP-LU next-hop prefixes from CE2 to the secondary IP address.
4. Forward packets based on next-hop: ASBR1 forwards traffic for CE2 via the AA tunnel based on the next-hop prefix.
5. Select preferred next-hop path: If two paths are learned (RSVP-TE and regular), the regular next-hop path is chosen.
6. Provide failure protection with FRR: Fast Reroute (FRR) protects against link and node failures during forwarding.

Result

Traffic between CE1 and CE2 is forwarded reliably and efficiently, with automatic protection against failures to maintain continuous network service.

Guidelines for BGP LU over RSVP-TE

- Do not configure BGP-LU over RSVP-TE simultaneously with BGP-LU (over NH) and Class-based forwarding (CBF). You must disable BGP-LU and CBF before enabling BGP-LU over RSVP-TE to avoid error messages.
- BGP-LU over RSVP-TE is not supported on Q100-based line cards.
- BGP-LU SR-TE is not supported.
- L3VPN, 6PE, and 6VPE services are not supported with BGP-LU over RSVP-TE.
- Use LDP or SR as the transport underlay. Do not use Traffic Engineering (TE) as the transport underlay.
- Reaching ASBR (BGP next-hop) through both regular next-hop and RSVP-TE is not supported.

Configure BGP-LU over RSVP-TE

Use this task when you need to deploy BGP-LU over RSVP-TE, optimize path selection with Forwarding-Adjacency (FA) and Autoroute Announce (AA) tunnels, and maintain protection against link or node failures.

Enable routers to forward BGP labeled unicast (BGP-LU) traffic through RSVP-TE tunnels, allowing you to select optimal tunnel paths and ensure traffic continuity with fast reroute (FRR) protection.

Before you begin

- Verify existing BGP-LU and Class-Based Forwarding (CBF) configurations are not active.
- Ensure you have appropriate router access and privileges.

Follow these steps to configure BGP-LU over RSVP-TE:

Procedure

Step 1 Disable BGP-LU and CBF configurations:

Example:

```
Router(config)# no hw-module profile cef bgplu enable
Router(config)# no hw-module profile cef cbf enable
```

Step 2 Enable BGP-LU over RSVP-TE and optionally increase tunnel capacity:

Example:

```
Router(config)# hw-module profile cef bgplu-over-rsvpte enable
Router(config)# hw-module profile cef te-tunnel highscale-no-ldp-over-te
```

Step 3 Configure the loopback interface:

Example:

```
Router(config)# interface Loopback1001
Router(config-if)# ipv4 address 10.10.10.10 255.255.255.255
Router(config-if)# exit
```

Step 4 Configure the tunnel interface:

Example:

```
Router(config)# interface tunnel-te1
Router(config-if)# ipv4 unnumbered Loopback0
Router(config-if)# autoroute announce
Router(config-if)# exit
Router(config)# destination 10.10.10.11
Router(config)# path-option 1 dynamic
```

Step 5 Configure the BGP router and address families:

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 10.10.10.10
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp)# allocate-label all unlabeled-path
Router(config-bgp)# exit
Router(config-bgp)# address-family ipv6 unicast
Router(config-bgp)# exit
```

Step 6 Configure the BGP neighbor:

Example:

```
Router(config-bgp)# neighbor 10.0.0.1
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family ipv4 labeled-unicast
Router(config-bgp-nbr-af)# route-policy PASS-ALL in
Router(config-bgp-nbr-af)# route-policy PASS-ALL out
Router(config-bgp-nbr-af)# next-hop-self
Router(config-bgp-nbr-af)# exit
```

Step 7 Configure MPLS LDP:

Example:

```
Router(config)# mpls ldp
Router(config-ldp)# router-id 10.1.1.1
Router(config-ldp)# interface tunnel-te1
Router(config-ldp)# exit
```

Step 8 Reload the router to apply the hardware module profile commands.

BGP-LU over RSVP-TE is now active, optimal tunnel paths are selected, and traffic is protected by fast reroute capabilities.

What to do next

Continue monitoring network performance and verify reroute operation during link or node failure events.

Verify BGP labeled unicast over RSVP-TE

Verify that BGP labeled unicast traffic forwards correctly through RSVP-TE tunnels. This ensures your configuration works as intended, enabling optimized traffic transport and fast reroute protection for reliable network performance.

Procedure

Step 1 Verify the configuration:

Example:

```
Router# show running configuration

Router configuration:
!
hw-module profile cef bgplu-over-rsvpte enable
!
router bgp 200
 nsr
 bgp router-id 10.1.1.1
 mpls activate
  interface Bundle-Ether10
  interface Bundle-Ether40
  interface Bundle-Ether100
  interface Bundle-Ether101
  interface HundredGigE0/0/0/22
 !
 bgp graceful-restart
 ibgp policy out enforce-modifications
 address-family ipv4 unicast
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy INSTALL_BACKUP
  network 10.1.1.5/32
  allocate-label all unlabeled-path
 !
 neighbor 10.1.4.1
  remote-as 200
  bfd fast-detect
  bfd multiplier 3
  bfd minimum-interval 100
  update-source Loopback0
  address-family ipv4 labeled-unicast
   next-hop-self
   soft-reconfiguration inbound always
   !
 neighbor 10.1.5.1
  remote-as 200
  bfd fast-detect
  bfd multiplier 3
  bfd minimum-interval 100
  update-source Loopback0
  address-family ipv4 labeled-unicast
   next-hop-self
   soft-reconfiguration inbound always
  !
  !
 neighbor 10.1.6.1
  remote-as 200
  bfd fast-detect
  bfd multiplier 3
  bfd minimum-interval 100
  address-family ipv4 labeled-unicast
   next-hop-self
   route-policy PASS-ALL in
   route-reflector-client
```

```
   route-policy PASS-ALL out
  !
Enabling LDP (to assign labels to the tunnel):

mpls ldp
 router-id 10.1.1.1
 address-family ipv4
  label
   local
    allocate for ldp-acl
   !
  !
router isis core
 is-type level-2-only
 net 49.1111.0000.0001.00
 nsr
 nsf cisco
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  mpls traffic-eng igp-intact
 !
 address-family ipv6 unicast
  metric-style wide
  maximum-paths 64
 !
 interface Bundle-Ether40
  circuit-type level-2-only
  point-to-point
  address-family ipv4 unicast
   metric 10
  !
  address-family ipv6 unicast
   metric 10
  !
   interface Bundle-Ether100
  circuit-type level-2-only
  point-to-point
  address-family ipv4 unicast
   metric 10
  !
  address-family ipv6 unicast
   metric 10
  !
 interface Bundle-Ether101
  circuit-type level-2-only
  point-to-point
  address-family ipv4 unicast
   metric 10
  !
  address-family ipv6 unicast
   metric 10
  !
Tunnel Configuration:

interface tunnel-te141
 description PE1-PE4
 ipv4 unnumbered Loopback0
 signalled-bandwidth 1000000
 autoroute announce
 !
 destination 10.1.4.1
```

```
 fast-reroute
 path-protection
 !
 path-option 1 explicit name R1-R4-141
!
interface tunnel-te142
 description PE1-PE4
 ipv4 unnumbered Loopback0
 shutdown
 signalled-bandwidth 1000000
 autoroute announce
 !
 destination 10.1.4.1
 fast-reroute
 path-option 1 explicit name R1-R4-142
!
interface tunnel-te13641
 ipv4 unnumbered Loopback0
 signalled-bandwidth 1000000
 autoroute announce
 !
 destination 10.1.4.1
 path-option 1 explicit name R1-R3-R6-R4-Phy protected-by 2
 path-option 2 explicit name R1-R3-R6-R4-Bundle
!
!

mpls traffic-eng
 interface Bundle-Ether10
 !
 interface Bundle-Ether100
  backup-path tunnel-te 13641
 !
 interface Bundle-Ether101
  backup-path tunnel-te 13641
```

Step 2 Verify the details of route paths:

Example:

```
Router# show cef 209.165.200.225/27

Tue Jun  6 13:59:39.649 UTC
201.1.1.10/32, version 838761, internal 0x5000001 0x40 (ptr 0xb6848370) [1], 0x600 (0xb67bc1d8),
0xa08 (0xbbc3c0d8)
 Updated Jun  6 13:56:34.879
 Prefix Len 32, traffic index 0, precedence n/a, priority 4
  gateway array (0xc020eac8) reference count 3, flags 0x100078, source rib (7), 0 backups
                [2 type 5 flags 0x441 (0xc1807b38) ext 0x0 (0x0)]
  LW-LDI[type=5, refc=3, ptr=0xb67bc1d8, sh-ldi=0xc1807b38]
  gateway array update type-time 1 Jun  6 13:56:34.879
 LDI Update time Jun  6 13:56:34.879
 LW-LDI-TS Jun  6 13:56:34.879
   via 10.1.4.1/32, 60047 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x97518b90 0x0]
    recursion-via-/32
    next hop 10.1.4.1/32 via 24000/0/21
     local label 36112
     next hop 10.1.4.1/32 tt141        labels imposed {ImplNull 34184}
     next hop 10.1.4.1/32 tt142        labels imposed {ImplNull 34184}
     next hop 10.1.4.1/32 tt13641      labels imposed {ImplNull 34184}
   via 10.1.5.1/32, 30045 dependencies, recursive, backup [flags 0x6100]
    path-idx 1 NHID 0x0 [0x97524fc0 0x0]
    recursion-via-/32
    next hop 10.1.5.1/32 via 24002/0/21
```

```
     local label 36112
     next hop 10.1.5.1/32 tt13651      labels imposed {ImplNull 39146}

    Load distribution: 0 (refcount 2)

    Hash  OK  Interface                 Address
    0     Y   recursive                 24000/0
```

Example:

```
Router# show route 10.1.4.1
Tue Jun  6 14:02:31.653 UTC

Routing entry for 10.1.4.1/32
  Known via "isis core", distance 115, metric 20, type level-2
  Installed Jun  6 13:59:07.013 for 00:03:24
  Routing Descriptor Blocks
    10.1.4.1, from 10.1.4.1, via tunnel-te141
      Route metric is 20
    10.1.4.1, from 10.1.4.1, via tunnel-te142
      Route metric is 20
    10.1.4.1, from 10.1.4.1, via tunnel-te13641
      Route metric is 20
  No advertising protos.
```

Example:

```
Router# show route summary
Wed May 31 17:47:01.203 UTC
Route Source                      Routes     Backup     Deleted     Memory(bytes)
connected                         536        2          0           116248
local                             539        0          0           116424
local LSPV                        1          0          0           216
local SMIAP                       1          0          0           216
application fib_mgr               0          0          0           0
static                            4          0          0           904
bgp 200                           48152      60         0           11936632
te-client                         0          0          0           0
isis core                         14056       534        0           4088288
dagr                              0          0          0           0
vxlan                             0          0          0           0
Total                             61364      596        0           16202240
```

Step 3 Verify the details of LSP tunnel:

Example:

```
Router# show mpls forwarding prefix 209.165.200.225/27
Tue Jun  6 14:00:17.601 UTC
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
36112  34184       209.165.200.225/27                10.1.4.1        0
       39146       209.165.200.225/27                10.1.5.1        0
```

Step 4 Verify the contents of the Fast Reroute (FRR) database:

Example:

```
show mpls traffic-eng fast-reroute database
Tue Jun  6 14:01:59.907 UTC
Tunnel head FRR information:
Tunnel       Out Intf : Label   FRR Intf : Label   Status
------------ ------------------ ------------------ -------
tt141        BE100:Pop          tt13641:Pop        Ready
tt142        BE101:Pop          tt13641:Pop        Ready
```

Step 5 Verify the forwarding information on tunnels:

Example:

```
Router# show mpls traffic-eng forwarding tunnel-id 141
Mon Jun  5 23:46:04.961 UTC
P2P tunnels:

Tunnel ID                 Ingress IF     Egress IF      In lbl  Out lbl        Backup
------------------------- -------------- -------------- ------- -------------- -------
10.1.1.1 141_10                        -          BE100 81920   3              tt13641
Displayed 1 tunnel heads, 0 label P2P rewrites
Displayed 0 tunnel heads, 0 label P2MP rewrites
```

Step 6 Verify the utilization of banks in the NPU resources:

Example:

```
Router# show grid pool 2 bank 13
Wed May 31 17:46:56.848 UTC

Bank Ptr                     : 0x308d069d38
Bank ID                      : 13
Pool                         : GLIF (id 2)
Bank Start                   : 530295
Bank End                     : 589823
Max Bank Size                : 59529
Max Resource Pages           : 1861
Available resource IDs       : 11375 (19.108% free)
Bank statistics:                          Success      Error         (since last clear)
  Resource IDs reserved                     51728          0           51728          0
  Resource IDs returned                      3574          0            3574          0
Client                       : lsd
  Resource IDs reserved                         2          0               2          0
  Resource IDs returned                         0          0               0          0
current usage                : 2
Client                       : rib-v4
  Resource IDs reserved                     51726          0           51726          0
  Resource IDs returned                      3574          0            3574          0
```

BGP labeled unicast version 6

BGP labeled unicast version 6 is a routing feature that

- enable MPLS transport between Provider Edge (PE) routers separated by multiple IGP boundaries (intra-AS) or autonomous systems (inter-AS)
- use autonomous system border routers (ASBRs) to advertise PE loopback prefixes and their MPLS label bindings, and
- support iBGP between area border routers (ABRs) and eBGP between ASBRs for route exchange.

Key considerations for BGP LU and VPN services across autonomous systems

Multihop eBGP can be used between PEs in different autonomous systems to exchange VPN routes. Services such as 6PE operate between PEs that have BGP Labeled Unicast connectivity.

BGP LU feature reduces scale requirements for IGP labeled prefixes and adjacencies. To prevent potential scaling issues when BGP Labeled Unicasts are not configured, you must configure the **hw-module** command before enabling BGP LU and restart the router for the changes to take effect.

BGP LU v6 extends this functionality to IPv6 networks, allowing you to apply the same MPLS transport principles over IPv6.

Table 41: Feature History Table

Feature Name	Release Information	Feature Description
BGP Labeled Unicast Version 6	Release 7.3.16	This feature extends the BGP Labeled Unicast (LU) functionality over IPv6. This feature provides connectivity between PEs to run services, such as L3VPN and 6PVE. This feature allows the PEs to transport traffic across autonomous systems (AS) boundaries. BGP LU allows you to transport MPLS traffic across IGP boundaries. By advertising loopbacks and label bindings across IGP boundaries routers communicate with other routers in remote areas that do not share the same local IGP.

Limitations for BGP labeled unicast version 6

You must understand the limitations of BGP Labeled Unicast Version 6 (BGP LU v6) to ensure proper deployment and avoid unsupported configurations that may cause network issues.

- 6VPE over BGP LU is not supported
- Inter-Address Family Identifier (AFI) is not supported.
- BGP PIC core feature is not supported.
- Coexistence of 6PE with the same neighbor is not supported.
- Coexistence of BGP LU v6 IPv6 unicast address family is not supported.
- VPNv6 over BGP LU v6 and Inter-AS Option-C with BGP LU v6 are not supported.
- Link-local addresses are not supported.
- Cases where BGP LU itself is the transport are not supported.
- Carrier Supporting Carrier version 6 is not supported.

Configure BGP labeled unicast version 6

Enable and configure BGP LU v6 on the hardware module, set up the BGP router and router ID, configure IPv6 unicast address family with route redistribution and label allocation, establish BGP neighbor and session parameters, and apply route policies for IPv6 labeled-unicast.

Use this task to enable labeled IPv6 routing with BGP LU v6, ensuring proper neighbor configuration and route policy application for efficient labeled-unicast forwarding.

Before you begin

Verify hardware module supports BGP LU and that you have the necessary router and neighbor information.

Follow these steps to enable and configure BGP LU v6:

Procedure

Step 1 Enable BGP labeled unicast on the hardware module

Example:

```
Router(config)# hw-module profile cef bgplu enable
```

Step 2 Configure the BGP router and router ID:

Example:

```
Router(config)# router bgp 1
Router(config-bgp)# bgp router-id 2001:DB8::1
```

Step 3 Configure the IPv6 unicast address family and redistribute connected routes with label allocation:

Example:

```
Router(config-bgp)# address-family ipv6 unicast
Router(config-bgp-af)# redistribute connected route-policy set-lbl-idx
Router(config-bgp-af)# allocate-label all
Router(config-bgp-af)# exit
```

Step 4 Configure the BGP neighbor and session parameters:

Example:

```
Router(config-bgp)# neighbor 2001:DB8::2
Router(config-bgp)# remote-as 1
Router(config-bgp)# update-source Loopback 0
```

Step 5 Configure the IPv6 labeled-unicast address family and apply route policies:

Example:

```
Router(config-bgp)# address-family ipv6 labeled-unicast
Router(config-bgp)# route-policy pass-all in
Router(config-bgp)# route-policy pass-all out
Router(config-bgp)# commit
```

Step 6 Verify the configuration:

Example:

```
Router# show running-config
hw-module profile cef bgplu enable
router bgp 1
 bgp router-id 2001:DB8::1
 address-family ipv6 unicast
  redistribute connected route-policy set-lbl-idx
  allocate-label all
  exit
 neighbor 2001:DB8::2
 remote-as 1
 update-source Loopback 0
  address-family ipv6 labeled-unicast
```

```
   route-policy pass-all in
   route-policy pass-all out
```

Step 7 Use the **show hw-module profile cef** to verify that the BGP LU has been configured.

Example:

```
Router# show hw-module profile cef
Thu Jun 17 00:06:32.974 UTC
-------------------------------------------------------------------------------------
Knob                          Status        Applied   Action
-------------------------------------------------------------------------------------
BGPLU                        Configured      Yes       None
LPTS ACL                     Unconfigured    Yes       None
Dark Bandwidth               Unconfigured    Yes       None
MPLS Per Path Stats          Unconfigured    Yes       None
Tunnel TTL Decrement         Unconfigured    Yes       None
High-Scale No-LDP-Over-TE    Unconfigured    Yes       None
IPv6 Hop-limit Punt          Unconfigured    Yes       None
IP Redirect Punt             Unconfigured    Yes       None
```

Step 8 Use the **show cef ipv6** command to verify the details of route paths along with the BGP and transport label information.

Example:

```
Router# show cef ipv6 192:168:9::80/128

Wed Jun 16 07:42:04.789 UTC
192:168:9::80/128, version 27, internal 0x5000001 0x40 (ptr 0x93f2d478) [1], 0x0 (0x93ef6cc0), 0xa08
 (0x9460a8a8)
 Updated Jun 16 07:36:00.189
 Prefix Len 128, traffic index 0, precedence n/a, priority 4, encap-id 0x1001000000001
   via 10:0:1::51/128, 3 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x94720660 0x0]
    recursion-via-/128
    next hop 10:0:1::51/128 via 16061/0/21
     next hop fe80::7af8:c2ff:fee4:20c0/128 Hu0/0/0/27   labels imposed {16061 25001}
/*
16061 - Transport Label
25001 – BGP Label
*/
```

Step 9 Use the **show bgp ipv6 unicast labels** to verify the BGP LU version 6 routes and BGP label information in BGP process.

Example:

```
Router# show bgp ipv6 unicast labels
Wed Jun 16 07:34:58.968 UTC
BGP router identifier 10.0.1.50, local AS number 1
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800000   RD version: 6
BGP main routing table version 6
BGP NSR Initial initsync version 3 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
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        Rcvd Label      Local Label
*> 192:168::/64        192:168:1::70   nolabel         24006
*>i192:168:9::80/128   10:0:1::51      25001           nolabel
```

```
Processed 2 prefixes, 2 paths
```

BGP LU v6 is enabled and configured, allowing labeled IPv6 routing with proper neighbor and route policy settings.

What to do next

Reload the router for the **hw-module profile cef bgplu enable** command to take effect.

BGP labeled unicast MPLS IP POP

BGP Labeled Unicast MPLS IP POP is a routing feature that

- enables efficient forwarding of IPv4 unicast traffic using labeled routes
- uses BGP to distribute labeled prefixes with implicit NULL labels, and
- applies only the transport LDP label on packets before forwarding them into the MPLS core.

This mechanism optimizes label stacking and improves forwarding efficiency in MPLS networks.

Table 42: Feature History Table

Feature Name	Release Information	Feature Description
BGP Labeled Unicast MPLS IP POP Support	Release 7.3.1	This feature is based on the BGP labeled Unicast feature. This feature enables a router to send unicast traffic to the destination from BGP labeled unicast using implicit NULL label. Implicit null label avoids adding or removing rewrites for neighbor flaps.

BGP labeled unicast with implicit NULL label and transport LDP label forwarding in MPLS core networks

- IP POP Support allows you to forward IP packets with minimal label stacking, improving network efficiency.
- For example, in a topology where client A sends IPv4 unicast traffic to client B, IP POP Support enables the Provider Edge (PE) router to add only the transport LDP label on top of the IP packet before forwarding it into the MPLS core.
- This reduces complexity compared to traditional MPLS forwarding, which may use multiple stacked labels.
- Understanding IP POP Support helps you design and troubleshoot MPLS networks that optimize label usage and forwarding performance.

Configure BGP labeled unicast on MPLS IP pop Support

Enable and configure BGP labeled unicast on PE1 and PE3 to optimize IPv4 labeled-unicast forwarding by reducing label overhead and simplifying MPLS label stacking.

This configuration allows PE1 to learn destination prefixes from PE3 via BGP labeled unicast using implicit NULL labels. PE1 then applies only the transport LDP label on IPv4 packets before forwarding them into the MPLS core, enhancing forwarding efficiency.

Before you begin

- Ensure hardware modules on PE1 and PE3 support BGP labeled unicast and the hardware module profile can be enabled.
- Confirm PE1 and PE3 have proper connectivity and reachability, including configured loopback interfaces for BGP update sources.
- Make sure BGP is operational on both routers with correct autonomous system numbers.

Configure PE1:

Procedure

Step 1 Enable BGP labeled unicast on the hardware module:

Example:

```
Router(config)# hw-module profile bgplu enable
```

Step 2 Configure BGP router and enable features

Example:

```
Router(config)# router bgp 200
Router(config-bgp)# nsr
Router(config-bgp)# bgp router-id 192.168.70.24
Router(config-bgp)# bgp graceful-restart
Router(config-bgp)# ibgp policy out enforce-modifications
```

Step 3 Configure IPv4 unicast address family with maximum paths and label allocation:

Example:

```
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# maximum-paths ibgp 8
Router(config-bgp-af)# network 101.101.1.0/24
Router(config-bgp-af)# network 101.101.2.0/24
Router(config-bgp-af)# allocate-label all
Router(config-bgp-af)# exit
```

Step 4 Configure BGP neighbor and session parameters:

Example:

```
Router(config-bgp)# neighbor 10.3.3.3
Router(config-bgp)# remote-as 200
Router(config-bgp)# update-source Loopback0
```

Step 5 Configure IPv4 labeled unicast address family and set the next-hop:

Example:

```
Router(config-bgp)# address-family ipv4 labeled-unicast
Router(config-bgp)# next-hop self
```

Step 6 Verify the configuration on the PE1 router. The output should show the bgplu profile enabled and correct BGP labeled unicast settings.

Example:

```
Router# show running-config
hw-module profile bgplu enable
router bgp 200
 nsr
 bgp router-id 192.168.70.24
 bgp graceful-restart
 ibgp policy out enforce-modifications
 address-family ipv4 unicast
  maximum-paths ibgp 8
  network 101.101.1.0/24
  network 101.101.2.0/24
  allocate-label all
!
 neighbor 10.3.3.3
 remote-as 200
 update-source Loopback0
 address-family ipv4 labeled-unicast
 next-hop self
```

Step 7 Verify the configuration on the PE2 router. The output should show the bgplu profile enabled and correct BGP labeled unicast settings.

Example:

```
Router# show running-config
hw-module profile bgplu enable
router bgp 200
 nsr
 bgp router-id 192.168.70.25
 bgp graceful-restart
 ibgp policy out enforce-modifications
 address-family ipv4 unicast
  maximum-paths ibgp 8
  network 103.101.1.0/24
  network 103.101.2.0/24
  allocate-label all
!
neighbor 10.1.1.1
remote-as 1
update-source Loopback0
address-family ipv4 labeled-unicast
next-hop self
```

Step 8 Verify if the feature has been configured.

Example:

```
Router# show bgp ipv4 unicast labels
  Network           Next Hop    Rcvd Label      Local Label
*>i103.101.1.0/24   10.3.3.3         3             24006
```

Step 9 Restart the router so hardware module configuration takes effect.

PE routers use BGP labeled unicast with implicit NULL labels and apply only the transport LDP label before forwarding IPv4 packets into the MPLS core, resulting in improved forwarding efficiency.

What to do next

Monitor BGP and MPLS operations to ensure stable label distribution and end-to-end forwarding.

Convergence for BGP labeled unicast PIC edge

A convergence for BGP labeled unicast PIC edge is a routing resiliency feature that

- allows you to store both primary and backup paths in the Routing Information Base (RIB), Forwarding Information Base (FIB), and Cisco Express Forwarding,
- enables the router to switch immediately to the backup path when a failure is detected, and
- provides subsecond convergence and fast failover for BGP-labeled prefixes.

Table 43: Feature History Table

Feature Name	Release Information	Feature Description
Convergence for BGP Labeled Unicast PIC Edge	Release 7.7.1	This feature improves the convergence time of BGP labeled unicast (LU) routes to subseconds when an ingress provider edge router fails or loses PE router connectivity, and another PE router needs to be connected. This feature minimizes traffic drops when the primary paths fail for the BGP LU routes.

Table 44: BGP LU PIC edge vs. standard BGP LU

	BGP LU PIC edge	Standard BGP LU
Backup path preprogrammed	Yes	No
Failover speed	Subsecond	Slower, may take seconds
Service disruption	Minimal	Possible disruption

Note You must ensure that your edge iBGP devices, such as ingress PEs and ASBRs, support BGP PIC and receive backup BGP next hops for PIC edge features to function.

How BGP labeled unicast PIC edge ensures fast convergence

This process applies to provider networks using BGP labeled unicast (LU) prefix independent convergence (PIC) edge to ensure rapid failover and maintain uninterrupted traffic during path or router failures.

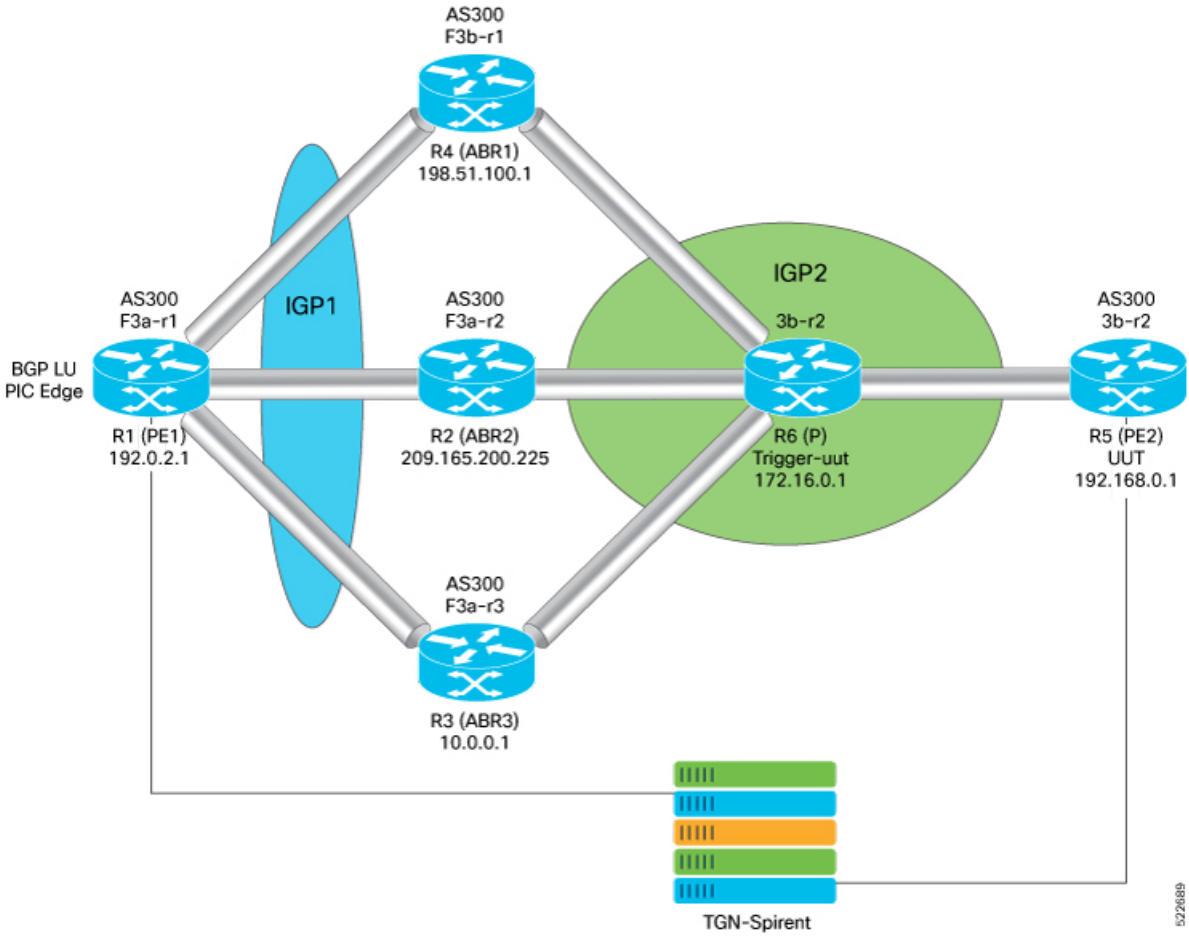
Summary

- PE1 (Provider Edge Router): Enables BGP LU PIC edge, learns BGP LU prefixes, programs primary and backup paths, detects failures, and switches traffic to the backup path when needed.
- PE2 (Remote Provider Edge Router): Provides BGP LU prefixes to PE1 and serves as a backup or alternate next hop.
- ABR1, ABR2, ABR3 (Area Border Routers): Serve as primary and backup transit paths for traffic between PE1 and PE2; if one ABR fails, another ABR takes over as the active path.

This process enables rapid failover in provider networks using BGP LU PIC edge by preprogramming primary and backup paths, ensuring fast, automatic rerouting and uninterrupted traffic during failures.

Workflow

Figure 24: BGP labeled unicast prefix independent convergence edge



These stages describe how BGP labeled unicast (LU) PIC edge enables fast convergence and failover in the provider network work.

1. Enable BGP LU PIC edge on the provider edge router (PE1).
2. PE1 learns BGP LU prefixes from the remote provider edge router (PE2).

3. PE1 programs both the primary and backup paths through area border routers (ABR1, ABR2, and ABR3) into the forwarding information base (FIB).
4. Program PE2 as the backup or alternate next hop in the FIB of PE1.
5. Under normal conditions, PE1 forwards traffic to the customer edge (CE) device through the primary ABR.
6. If the primary ABR fails, PE1 immediately activates the preprogrammed backup path and forwards traffic through an alternate ABR.
7. When PE1 detects that PE2 is not reachable via the primary ABR, it switches the BGP next hop for the prefix to the alternate ABR without delay.
8. The switchover to the backup path occurs in less than a second, even if multiple BGP prefixes are updated at once.

Result

This process ensures subsecond convergence and uninterrupted traffic flow during ABR or path failures in the network.

Supported features and limitations of BGP LU PIC edge

Supported features:

- BGP LU PIC edge supports BGP multipaths, allowing routers to install multiple internal and external BGP paths in the forwarding table.
- Multiple paths enable BGP to load balance traffic across several links.

Limitation::

- Convergence time is independent of the BGP LU route scale.

Configure convergence for BGP labeled unicast prefix independent convergence edge

Configure BGP LU PIC edge and multipath to enable fast convergence and backup path installation.

Use this task when you need to configure BGP labeled unicast PIC edge and multipath on Cisco routers to ensure fast convergence and automatic traffic failover in service provider or large enterprise networks.

Before you begin

- Ensure you have access to the router prompt in configuration mode.
- Verify that BGP is enabled and address families are configured as needed.

Procedure

Step 1 Configure BGP labeled unicast and attach a route-policy to BGP address families.

Example:

Apply the route policy to the BGP address family.

```
Router(config)# route-policy BGP-PIC-EDGE
Router(config-rpl)# set path-selection backup 1 install
Router(config-rpl)# end-policy
```

Apply the route policy to the BGP address family.

```
Router(config)# router bgp <ASN>
Router(config-bgp)# bgp router-id <Router-ID>
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# additional-paths receive
Router(config-bgp-af)# additional-paths send
Router(config-bgp-af)# additional-paths selection route-policy BGP-PIC-EDGE
```

Step 2 Configure BGP labeled unicast multipath and attach a route-policy to BGP address families:

Example:

Define the multipath route policy.

```
Router(config)# route-policy BGP-PIC-EDGE-MULTIPATH
Router(config-rpl)# set path-selection backup 1 install multipath-protect
Router(config-rpl)# end-policy
```

Apply the multipath route policy to the BGP address family.

```
Router(config)# router bgp <ASN>
Router(config-bgp)# bgp router-id <Router-ID>
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# maximum-paths ibgp 2
Router(config-bgp-af)# additional-paths receive
Router(config-bgp-af)# additional-paths send
Router(config-bgp-af)# additional-paths selection route-policy BGP-PIC-EDGE-MULTIPATH
```

Step 3 Use the **show running-config** to verify if the feature has been configured.

Example:

```
Router# show running-config
route-policy BGP-PIC-EDGE
 set path-selection backup 1 install
 end-policy
router bgp 200
 bgp router-id 192.168.1.0
 address-family ipv4 unicast
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy BGP-PIC-EDGE

route-policy BGP-PIC-EDGE-MULTIPATH
 set path-selection backup 1 install multipath-protect
 end-policy
router bgp 200
 bgp router-id 192.168.1.0
 address-family ipv4 unicast
  maximum-paths ibgp 2
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy BGP-PIC-EDGE-MULTIPATH
```

Step 4 Use the **show cef** to verify that the backup path is established.

Example:

```
Router# show cef 192.0.2.1/32

192.168.0.0/32, version 31, internal 0x5000001 0x40 (ptr 0x901d2370) [1], 0x0 (0x90d2beb8), 0xa08
(0x91c74378)
 Prefix Len 32, traffic index 0, precedence n/a, priority 4
   via 203.0.113.1/32, 3 dependencies, recursive [flags 0x6000]  << Primary Path
    path-idx 0 NHID 0x0 [0x90319650 0x0]
    recursion-via-/32
    next hop 192.51.100.1/32 via 24006/0/21
    next hop 209.165.200.225/32 Hu0/0/0/25   labels imposed {24002 24000}
    next hop 10.0.0.1/32 Hu0/0/0/26   labels imposed {24002 24000}
   via 203.0.113.2/32, 2 dependencies, recursive, backup [flags 0x6100]  << Backup Path
    path-idx 1 NHID 0x0 [0x903197b8 0x0]
    recursion-via-/32
    next hop 209.165.200.225/32 via 24005/0/21
    next hop 192.51.100.1/32 Hu0/0/0/25   labels imposed {24001 24000}
    next hop 10.0.0.1/32 Hu0/0/0/26   labels imposed {24001 24000}
```

BGP labeled unicast PIC edge and multipath are configured, enabling the router to install backup paths and achieve fast, automatic convergence and failover during link or path failures.

Exclusion of label allocation for non-advertised routes

An exclusion of label allocation for non-advertised routes is a label management feature that

- allows control of label allocation based on route advertisement status
- enables assignment of labels only to routes that are advertised to BGP peers, and
- prevents label allocation for routes that are not advertised.

Route Advertisement Control Using Community Attributes on PE-ASBRs

- This feature is configured on Provider Edge (PE) routers acting as autonomous system border routers (ASBRs).

  The community attribute is set to either no-advertise or no-export in route-policy configuration mode for routes that are not advertised to peers.
- After applying the community attribute in the route policy, the router does not allocate a label to those routes.

Guidelines for applying BGP community attributes

Apply the BGP community attributes as follows:

- Use `no-advertise` for both eBGP and iBGP routes.
- Use `no-export` only for eBGP routes.

Table 45: Feature History Table

Feature Name	Release Information	Feature Description
Exclusion of Label Allocation for Non-Advertised Routes	Release 7.10.1	We have enabled better label space management and hardware resource utilization by making MPLS label allocation more flexible. This flexibility means you can now assign these labels to only those routes that are advertised to their peer routes, ensuring better label space management and hardware resource utilization. Prior to this release, label allocation was done regardless of whether the routes being advertised. This resulted in inefficient use of label space.

Exclude label allocation for non-advertised routes

Prevent label allocation to routes that are not advertised to any BGP peer by using the no-advertise community.

Use this task when you need to configure BGP on a Provider Edge (PE) router so that non-advertised routes do not receive labels.

Before you begin

- Verify that BGP is already configured on the device.
- Identify the BGP neighbors and address families that require this configuration.
- Plan the route policies and community sets you need to implement.

\

Procedure

Step 1 Configure the community set for no-advertise.

Example:

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

Step 2 Configure a route policy to set the no-advertise community.

Example:

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

Step 3 Apply the route policy as an inbound policy to the BGP neighbor.

Example:

```
Router(config)# router bgp 1
Router(config-bgp)# neighbor 192.0.2.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy set-no-advertise in
Router(config-bgp-nbr-af)# route-policy pass_all out
Router(config-bgp-nbr-af)# commit
```

Step 4 Use the **show bgp vpnv6 unicast rd** command to verify if the community parameter is set to *no-advertise*.

Example:

```
Router# show bgp vpnv6 unicast rd 2001:DB8:0:ABCD::1
BGP routing table entry for 0:ABCD::1 Route Distinguisher: 2001:DB8
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               19207        19207
Paths: (1 available, best #1, not advertised to any peer)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local, (Received from a RR-client)
    192.0.2.254 from 192.0.2.1 (192.0.2.1)
      Received Label 16
      Origin IGP, metric 3, localpref 3, aigp metric 3, valid, internal, best, group-best,
import-candidate, not-in-vrf
      Received Path ID 0, Local Path ID 1, version 19207
Community: 1:1 no-advertise
      Extended community: Color:3333 RT:2001:DB8
      AIGP set by inbound policy metric
      Total AIGP metric 3
```

Steering of BGP control-plane traffic over IP paths

Steering of BGP control-plane traffic over IP paths is a traffic engineering feature that

- allows selection of an IP-only transport path for BGP control-plane traffic instead of using the default MPLS LSP
- separates BGP control-plane traffic from labeled and regular IP traffic, and
- reduces complexity and risk by isolating BGP session traffic from MPLS transport paths.

Steering BGP control-plane traffic over IP-only paths in MPLS networks

In a typical underlay network, the transport label-switched path (LSP) is established using MPLS protocols such as Segment Routing MPLS, Label Distribution Protocol (LDP), or Service Layer API. By default, the transport LSP carries all traffic—including labeled packets, IP packets, and BGP control-plane traffic—toward the underlay destination. Routing BGP control-plane traffic over MPLS LSPs can introduce operational complexity and risk, potentially leading to network instability.

With the steering feature, you can configure BGP control-plane traffic to use an IP-only path created by the IS-IS protocol. The MPLS path continues to determine BGP next hops for data-plane traffic, while the IP-only path is used exclusively for BGP control-plane packets.

Before you enable this feature, you create a new VRF to manage IP-only routing tables. After configuration, IS-IS generates an IP-only route entry in the Routing Information Base (RIB), which is then downloaded to the Forwarding Information Base (FIB) in the VRF. This separate VRF topology allows the router to resolve locally generated BGP control-plane traffic independently from the MPLS transport.

Table 46: Feature History Table

Feature Name	Release Information	Feature Description
Steering of BGP Control-Plane Traffic over IP Path	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 88-LC1-36EH+A8:B12 • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8212-48FH-M • 8711-32FH-M
Steering of BGP Control-Plane Traffic over IP Path	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100]) This feature support is now extended to the Cisco 8712-MOD-M routers.

Steering of BGP Control-Plane Traffic over IP Path	Release 24.2.11	You can now steer the BGP control-plane traffic through an IP-only transport path even when MPLS Link State Packets (LSPs) are configured for BGP neighbor reachability. This feature allows you to keep the BGP control-plane traffic independent of the data plane traffic, enabling you to have more granular control over your network traffic. The feature introduces these changes: CLI: New Commands: • **table ip-only activate vrf** • **tcp ip-only-preferred** Modified Commands: • The **distribute-list** command is modified with a new **ip-only** keyword. YANG Data Models: New XPaths for • Cisco-IOS-XR-clns-isis-cfg.yang • Cisco-IOS-XR-ipv4-bgp-cfg.yang • Cisco-IOS-XR-ip-rib-cfg.yang • Cisco-IOS-XR-um-router-bgp-cfg.yang • Cisco-IOS-XR-um-router-isis-cfg.yang (see GitHub, YANG Data Models Navigator)

Configure the router to steer BGP control-plane traffic over an IP-only path

Configure the router to direct BGP control-plane traffic over an IS-IS IP-only path instead of the default MPLS LSP.

Use this task when you want to separate BGP control-plane traffic from MPLS transport paths in an underlay network. Steering BGP control-plane packets over an IP-only path can help reduce complexity and improve the stability of BGP sessions in large-scale or high-availability environments.

Before you begin

- Confirm that BGP and IS-IS are enabled and configured.
- Identify the ASN, remote AS, loopback interface, and process IDs you need.

Procedure

Step 1 Configure a VRF for the IP-only path.

Example:

```
Router(config)# vrf ip_only
Router(config-vrf)# fallback-vrf default
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# exit
Router(config-vrf)# address-family ipv6 unicast
Router(config-vrf-af)# exit
```

Step 2 Activate the IP-only table in RIB configuration.

Example:

```
Router(config)# router rib
Router(config-rib)# table ip-only activate vrf ip_only
```

Step 3 Configure the BGP neighbor group to use IP-only steering.

Example:

```
Router(config)# router bgp <ASN>
Router(config-bgp)# neighbor-group ip-only
Router(config-bgp-nbrgrp)# remote-as <Remote-AS>
Router(config-bgp-nbrgrp)# update-source <Loopback-Interface>
Router(config-bgp-nbrgrp)# tcp ip-only-preferred
```

Step 4 (Optional) Configure prefix-list and distribute-list for IS-IS.

Example:

```
Router(config)# ipv4 prefix-list v4-host-only
Router(config-ipv4_pfx)# 10 permit 0.0.0.0/0 eq 32
Router(config-ipv4_pfx)# exit
Router(config)# router isis 1
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# distribute-list ip-only prefix-list v4-host-only in
```

Step 5 (Optional) Configure a route-policy for IP-only steering.

Example:

```
Router(config)# route-policy rpl-isis-ip-only
Router(config-rpl)# if not destination in (192.0.2.1 192.0.2.2 192.0.2.3) then
Router(config-rpl-if)# drop
Router(config-rpl-if)# else
Router(config-rpl-else)# pass
Router(config-rpl)# end-policy
Router(config)# router isis 1
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# distribute-list ip-only route-policy isis-ip-only in
```

Step 6 Use the **show running-config router rib** command to verify if the feature is enabled.

Example:

```
Router# show running-config router rib
Wed Mar 27 06:39:01.233 UTC
router rib
 table ip-only activate vrf ip_only
!
```

Step 7 Verify the IS-IS IP-only local RIB entries:

Example:

```
 Router# show isis route ip-only

Wed Jul 26 09:24:56.422 PDT
```

```
IS-IS 1 IPv4 Unicast routes

Codes: L1 - level 1, L2 - level 2, ia - interarea (leaked into level 1)
       df - level 1 default (closest attached router), su - summary null
       C - connected, S - static, R - RIP, B - BGP, O - OSPF
       E - EIGRP, A - access/subscriber, M - mobile, a - application
       i - IS-IS (redistributed from another instance)

Maximum parallel path count: 8

L2 10.2.1.0/24 [20/115]
     via 10.1.1.101, GigabitEthernet0/0/0/2, r101, Weight: 0
L2 10.3.1.0/24 [120/115]
     via 10.1.1.101, GigabitEthernet0/0/0/2, r101, Weight: 0
L2 10.4.1.0/24 [130/115]
     via 10.1.1.101, GigabitEthernet0/0/0/2, r101, Weight: 0
L2 10.1.0.101/32 [20/115]
     via 10.1.1.101, GigabitEthernet0/0/0/2, r101, Weight: 0
L2 10.1.0.102/32 [30/115]
     via 10.1.1.101, GigabitEthernet0/0/0/2, r101, Weight: 0
L2 10.1.0.103/32 [130/115]
     via 10.1.1.101, GigabitEthernet0/0/0/2, r101, Weight: 0
```

Step 8 Use the **show tcp detail pcb** command to verify that BGP is using the IP-only option and check the TCP session details for the neighbor.

Example:

```
Router# show tcp detail pcb 0x00007f733000d618 location 0/rP1/CPU0

Tue Dec 12 09:20:56.163 UTC

================================================================
Connection state is ESTAB, I/O status: 0, socket status: 0
Established at Tue Dec 12 07:25:24 2023

PCB 0x00007f733000d618, SO 0x7f733000d158, TCPCB 0x7f733000d8c8, vrfid 0x60000000,
Pak Prio: Medium, TOS: 192, TTL: 255, Hash index: 1575
Local host: 10.1.1.1, Local port: 179 (Local App PID: 24619)
Foreign host: 10.4.4.4, Foreign port: 50026
(Local App PID/instance/SPL_APP_ID: 24619/1/0)

Current send queue size in bytes: 0 (max 24576)
Current receive queue size in bytes: 0 (max 32768)  mis-ordered: 0 bytes
Current receive queue size in packets: 0 (max 0)

   Timer          Starts    Wakeups         Next(msec)
Retrans           1735         0               0
SendWnd             0          0               0
TimeWait            0          0               0
AckHold           1733      1668               0
KeepAlive           0          0               0
PmtuAger            0          0               0
GiveUp              0          0               0
Throttle            0          0               0
FirstSyn            0          0               0

   iss: 2670304720  snduna: 2670348690  sndnxt: 2670348690
sndmax: 2670348690  sndwnd: 32768       sndcwnd: 3720
   irs: 2277543107  rcvnxt: 2277587077  rcvwnd: 32331   rcvadv: 2277619845

SRTT: 232 ms,  RTTO: 300 ms,  RTV: 7 ms,  KRTT: 0 ms
minRTT: 0 ms,  maxRTT: 248 ms
```

```
ACK hold time: 200 ms, Keepalive time: 0 sec, SYN waittime: 30 sec
Giveup time: 0 ms, Retransmission retries: 0, Retransmit forever: FALSE
Connect retries remaining: 0, connect retry interval: 0 secs

State flags: none
Feature flags: Win Scale, Nagle, IP FIB TBLID OVERRIDE
Request flags: Win Scale

Datagrams (in bytes): MSS 1240, peer MSS 1240, min MSS 1240, max MSS 1240

Window scales: rcv 0, snd 0, request rcv 0, request snd 0
Timestamp option: recent 0, recent age 0, last ACK sent 0
Sack blocks {start, end}: none
Sack holes {start, end, dups, rxmit}: none
Socket options: SO_REUSEADDR, SO_REUSEPORT, SO_NBIO
Socket states: SS_ISCONNECTED, SS_PRIV, SS_BLOCKCLOSE, SS_BLOCKSND
Socket receive buffer states: SB_DEL_WAKEUP
Socket send buffer states: SB_DEL_WAKEUP
Socket receive buffer: Low/High watermark 1/32768
Socket send buffer   : Low/High watermark 2048/24576, Notify threshold 0
Socket misc info     : Rcv data size (sb_cc) 0, so_qlen 0,
                       so_q0len 0, so_qlimit 0, so_error 0
                       so_auto_rearm 1

PDU information:
 #PDU's in buffer: 0

FIB Lookup Cache:
  Lookup table: default ipv4 unicast (Table ID: 0xe0000001)
  Lookup done at Tue Dec 12 09:16:24 2023 (next lookup due on next protocol message on or after 78
sec)

  Lookup result:
    Matching table: default ipv4 unicast (Table ID: 0xe0000001)
    Outgoing interface: Bundle-Ether1 (IFH: 0xf000024)
    PD ctx:  size: 0 data: {}
    Num Labels: 0  Label Stack: {}
    Next HopID: 0
    VXLAN Encap String size: 0 data:
    VXLAN Next Hop IP size: 0 IP:

Num of peers with authentication info: 0
```

Step 9 Use the **show tcp statistics pcb** command to verify the number of IP-only packets per neighbor:

Example:

```
Router# show tcp statistics pcb 0x00007f733000d618 location 0/rP1/CPU0

Wed Mar 27 06:46:52.566 UTC

============================================================
 Statistics for PCB 0x7f1ca0008550, vrfid 0x60000000
Send:   0 bytes received from application
        0 segment instructions received from partner
        0 xipc pulses received from application
        0 packets sent to network (v4/v6 IO)
        3547 packets sent to network (NetIO)
        0 packets failed getting queued to network (v4/v6 IO)
        0 packets failed getting queued to network (NetIO)
        3217 ip-only-preferred packets sent to network
        0 write operations by application
        0 times armed, 0 times unarmed, 0 times auto-armed
        Last written at: Wed Mar 27 06:46:51 2024
```

```
Rcvd:   3584 packets received from network
        1791 packets queued to application
        1 packets failed queuing to application
        0 packets dropped due to minttl check
        0 send-window shrink attempts by peer ignored
        0 read operations by application
        0 times armed, 0 times unarmed, 0 times auto-armed
        Last read at: Wed Mar 27 06:46:51 2024
```

Verify the BGP control-plane IP-only steering configuration

Confirm that the router is configured to steer BGP control-plane traffic over an IP-only path, separating BGP control-plane traffic from MPLS transport.

Use this task to verify that your router's running configuration supports BGP control-plane IP-only steering using IS-IS and VRF-based routing..

Before you begin

Before you verify the BGP control-plane IP-only steering configuration, ensure the following:

- You have administrative or enable-level access to the router.
- Prefix lists, distribute lists, and route policies for IP-only steering have been configured as intended.
- BGP is enabled and properly configured, including neighbor groups and update sources.

Follow these steps to verify BGP control-plane IP-only steering configuration:

Procedure

Step 1 Use the **show rib tables** command to verify the status and details of the RIB tables on the router

Example:

```
Router# show rib tables
Wed Mar 27 06:39:58.319 UTC

Codes: N - Prefix Limit Notified, F - Forward Referenced
       D - Table Deleted, C - Table Reached Convergence

VRF/Table              SAFI  Table ID     PrfxLmt   PrfxCnt TblVersion  N F D C
default/default        uni   0xe0000000  10000000        21         43  N N N Y
ip_only/default        uni   0xe0000001  10000000        10         42  N N N Y
default-ip-only/defau  uni   0xe0000002  10000000         0          0  N N N Y
**iid/default          uni   0xe00007d9  10000000         0          0  N N N Y
default/default        multi 0xe0100000  10000000         0          0  N N N Y
```

Step 2 Use **show isis rib tables** command to verify the IS-IS routing tables present on the router.

Example:

```
Router# show isis rib tables
Wed Mar 27 06:40:58.587 UTC

IS-IS 100 Routing Tables
```

```
  ISIS routes       VRF/Table                SAFI   Table ID     State

IPv4 Unicast:
  default           default/default          uni    0xe0000000   enabled
  ip-only           ip_only/default          uni    0xe0000001   enabled
  multicast-intact  default/default          uni    0xe0100000   enabled

IPv6 Unicast:
  default           default/default          uni    0xe0800000   enabled
  ip-only           ip_only/default          uni    0xe0800001   enabled
  srv6              default/default          uni    0xe0800000   enabled
```

Step 3 Use the **show running-config** to display the running configuration.

Example:

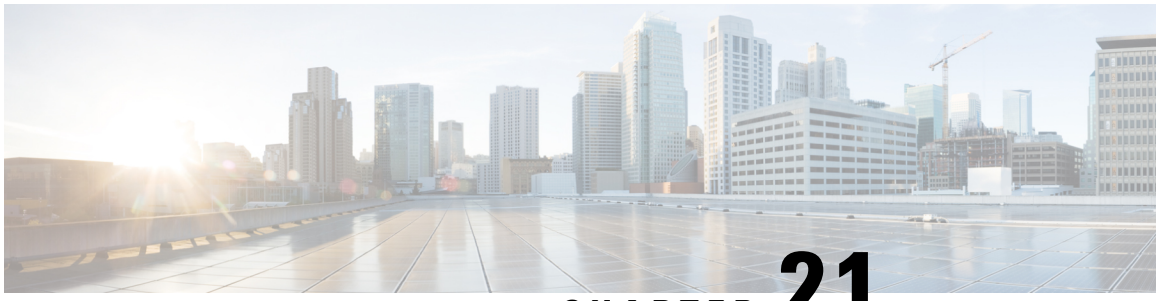
```
Router# show running-config
vrf ip_only
 fallback-vrf default
 address-family ipv4 unicast
 !
 address-family ipv6 unicast
 !
!
router rib
      table ip-only activate vrf ip_only
!
router bgp 140
 neighbor-group ip_only
  remote-as 100
  update-source Loopback99
  tcp ip-only-preferred
!
ipv4 prefix-list v4-host-only
  10 permit 0.0.0.0/0 eq 32
!
router isis 1
  address-family ipv4 unicast
 distribute-list ip-only prefix-list v4-host-only in
!
route-policy rpl-isis-ip-only
if not destination in (192.0.2.1 192.0.2.2 192.0.2.3) then
  drop
else
  pass
end-policy
!
router isis 1
  address-family ipv4 unicast
    distribute-list ip-only route-policy isis-ip-only in
!
!
```

Review the output to ensure the following configuration elements are present:

- A VRF named `ip_only` with appropriate address families.
- The RIB is configured to activate the `ip_only` VRF.
- The BGP neighbor group includes the `tcp ip-only-preferred` setting.
- Prefix-list and distribute-list settings for IS-IS, if required.

- Any custom route-policies for IP-only steering.

You have confirmed that BGP control-plane traffic is set to use an IP-only path, based on the elements present in the running configuration.



CHAPTER 21

BGP Flowspec

This chapter details BGP Flowspec, a routing feature enabling dynamic traffic filtering, policing, and automated threat mitigation, especially against DDoS attacks. It covers configuring client and server roles, verifying operations, and implementing advanced traffic redirection from global VRF to L3VPN or segment routing policies.

BGP flowspec

The BGP flowspec is a routing feature that

- dynamically distributes traffic filtering and policing rules
- enables granular control over network traffic, and
- automates threat mitigation across BGP-speaking routers.

You use BGP flowspec primarily to quickly and automatically respond to network threats, especially Distributed Denial-of-Service (DDoS) attacks. This feature allows you to deploy filtering rules rapidly across many routers, stopping attack traffic closer to its source. It provides granular control over traffic, letting you define precise matching criteria and actions. By automating rule deployment through BGP updates, you reduce manual configuration effort and ensure consistent policy enforcement across your network.

Table 47: Feature History Table

Feature Name	Release Information	Feature Description

BGP Flowspec	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) You can now rapidly deploy and propagate filtering and policing functionality across many BGP peer routers, which helps to mitigate the effects of a distributed denial-of-service (DDoS) attack on your network. This feature allows you to create detailed instructions for matching specific traffic flows based on various parameters (for example, IP addresses, ports, and packet specifics) and to define actions (such as dropping, policing, or redirecting the traffic) through BGP updates. This helps in effectively managing and mitigating unwanted traffic. *This functionality is now supported on: • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8212-48FH-M • 8711-32FH-M

Table 48: Feature History Table

Feature Name	Release Information	Feature Description
Scaling BGP Flowspec to 6000 Rules	Release 7.5.2	You can now assign 6000 BGP Flowspec rules for Cisco 8800 series routers and 3000 BGP Flowspec rules for Cisco 8100 and 8200 series routers. This feature thus provide enhanced mitigation against Distributed Denial-of-Service (DDoS) attacks. In earlier releases, you could assign 2000 BGP Flowspec rules. These are one dimensional scale numbers; the numbers vary based on other intersecting features like AccessList (ACL), Quality of Service (QoS), and Local Path Transport Switching (LPTS).

How BGP flowspec client-server controller models work

The BGP flowspec model operates with a controller (often a server) and clients (BGP speakers). The controller originates and injects the flowspec NLRI entries, while the clients receive these entries and program their hardware accordingly.

The BGP flowspec model operates with a controller (often a server) and clients (BGP speakers). The controller originates and injects the Flowspec NLRI entries, while the clients receive these entries and program their

hardware accordingly. This model is often visualized with the controller on the left-hand side (refer to the figure: BGP flowspec Controller) and the client on the right-hand side (refer to the figure: BGP flowspec Client), illustrating the flow of information.

Summary

The BGP Flowspec process describes how a controller defines and distributes traffic management rules to client routers, which then enforce these rules on network traffic. This ensures dynamic and granular control over data flows.

Workflow

Figure 25: BGP flowspec client

Figure 26: BGP flowspec controller

These stages describe the BGP flowspec client-server controller model:

1. Rule construction:
   - Actor: The controller
   - Action: The controller defines specific traffic flow rules. These rules include both detailed matching criteria (what traffic to identify) and the actions the network takes (what to do with that traffic). The controller is typically configured using commands to provide these entries for NLRI injection.
2. BGP propagation:
   - Actor: The controller
   - Action: The controller encodes these rules. It uses BGP NLRI for the matching criteria and BGP extended communities for the actions.
3. - Actor: The BGP flowspec-enabled controller
   - Action: The controller originates these rules and advertises them to its BGP peers, which function as clients or speakers.
4. Local enforcement:
   - Actor: The receiving BGP Flowspec Client routers
   - Action: The client routers install these rules into their local forwarding plane. The client receives the information, sends it to its internal flowspec manager, and configures the enhanced Policy-based Routing (ePBR) infrastructure, which in turn programs the underlying hardware.
5. Traffic processing:
   - Actor: The client router's active Layer 3 interfaces.
   - Action: Once installed and programmed in the applicable line cards, the interfaces start processing ingress traffic. They apply the specified actions to any traffic flows that match the defined rules.

Result

This process results in network devices actively identifying and applying predefined actions (such as filtering, policing, or redirection) to specific traffic flows in real-time, based on the rules distributed via BGP.

Restrictions for BGP flowspec

These are the specific restrictions for configuring of BGP flowspec. You should be aware of them when deploying and managing the BGP flowspec:

- Flowspec statistics are supported only when a policer rate limit is configured.
- The policer action scale is limited to a maximum of 128 per slice.
- Statistics for the Redirect action are supported only if a policer is attached; statistics are not supported for Redirect action alone.
- Redirects from a VRF to the default VRF are not supported.

Understanding these limitations helps you design and operate Flowspec policies effectively and avoid unsupported configurations.

Configure BGP flowspec on the client

This section provides configuration examples for a scenario where a BGP flowspec controller (server) with IP address 10.2.3.4 sends flowspec NLRI to a client with IP address 10.2.3.3. The NLRI contains matching criteria, and the Client processes traffic based on these criteria. Traffic is then dropped or accepted based on the configured criteria.

Before you begin

You must enable and configure the Border Gateway Protocol (BGP) routing process on both the client and server routers.

This task describes how you configure a BGP Flowspec client to receive and process flowspec NLRI from a BGP flowspec server.

Procedure

Step 1 Define a Virtual Routing and Forwarding (VRF) instance named vrf1 and set up import and export route targets for different address families.

Example:

```
Router(config)# router bgp 140
Router(config-bgp)# vrf vrf1
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# import route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# export route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# exit
```

```
Router(config-bgp-vrf)# address-family ipv4 flowspec
Router(config-bgp-vrf-af)# import route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# export route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# exit

Router(config-bgp-vrf)# address-family ipv6 unicast
Router(config-bgp-vrf-af)# import route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# export route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# exit

Router(config-bgp-vrf)# address-family ipv6 flowspec
Router(config-bgp-vrf-af)# import route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# export route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# exit

Router(config-bgp-vrf)# address-family vpnv4 flowspec
Router(config-bgp-vrf-af)# import route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# export route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# exit

Router(config-bgp-vrf)# address-family vpnv6 flowspec
Router(config-bgp-vrf-af)# import route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# export route-target
Router(config-bgp-vrf-af)# 101:2000
Router(config-bgp-vrf-af)# 201:2000
Router(config-bgp-vrf-af)# exit
```

Step 2 Configure BGP flowspec for various address families to allow local installation of rules on all interfaces.

Example:

```
Router(config)# flowspec
Router(config-flowspec)# address-family ipv4
Router(config-flowspec-af)# local-install interface-all
Router(config-flowspec-af)# exit
Router(config-flowspec)# address-family ipv6
Router(config-flowspec-af)# local-install interface-all
Router(config-flowspec-af)# exit
Router(config-flowspec)# address-family vpnv4
Router(config-flowspec-af)# local-install interface-all
Router(config-flowspec-af)# exit
Router(config-flowspec)# address-family vpnv6
Router(config-flowspec-af)# local-install interface-all
Router(config-flowspec-af)# exit
```

Step 3 Configure route policies to accept all routes (pass-all) and to reject all routes (drop-all).

Example:

```
Router(config)# route-policy pass-all
Router(config)# pass
Router(config)# end-policy

Router(config)# route-policy drop-all
Router(config)# drop
Router(config)# end-policy
```

Step 4 Configure the BGP process and define the neighbor relationship with the Flowspec server (10.2.3.4), applying the defined route policies for inbound and outbound Flowspec NLRI.

Example:

```
Router(config)# router bgp 1
Router(config-bgp)# nsr
Router(config-bgp)# bgp router-id 10.2.3.3
Router(config-bgp)# address-family ipv4 flowspec
Router(config-bgp-af)# exit
Router(config-bgp)# address-family ipv6 flowspec
Router(config-bgp-af)# exit

Router(config-bgp)# address-family vpnv4 flowspec
Router(config-bgp-af)# exit
Router(config-bgp)# address-family vpnv6 flowspec
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.2.3.4
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# address-family ipv4 flowspec
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy drop-all out
Router(config-bgp-af)# exit
Router(config-bgp-nbr)# address-family ipv6 flowspec
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy drop-all out
Router(config-bgp-nbr-af)# exit

Router(config-bgp-nbr)# address-family vpnv4 flowspec
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy drop-all out
Router(config-bgp-af)# exit
Router(config-bgp-nbr)# address-family vpnv6 flowspec
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy drop-all out
Router(config-bgp-af)# exit

Router(config-bgp-nbr)# update-source Loopback0
```

Step 5 Disable BGP flowspec.

Example:

```
Router(config)# interface bundle-ether 3.1
Router(config-subif)# ipv4 flowspec disable
Router(config-subif)# ipv6 flowspec disable
```

This configuration establishes a VRF instance named `vrf1` within BGP AS 140, defining import and export route targets for IPv4, IPv6, and VPN address families, enabling its participation in an MPLS VPN environment. Additionally, it configures the router as a BGP flowspec client that locally installs received flowspec rules on

all interfaces, accepting rules from neighbor 10.2.3.4 (a flowspec server) while preventing the advertisement of its own flowspec rules.

Configure BGP flowspec on the server

This task describes how you configure a BGP Flowspec server to define and advertise Flowspec NLRI to a BGP flowspec client.

-

Before you begin

-

Procedure

Step 1 Configure route policies to accept all routes (pass-all) and to reject all routes (drop-all).

Example:

```
Router(config)# route-policy pass-all
Router(config)# pass
Router(config)# end-policy

Router(config)# route-policy drop-all
Router(config)# drop
Router(config)# end-policy
```

Step 2 Configure the BGP process and define the neighbor relationship with the flowspec client (10.2.3.3), applying the defined route policies for inbound and outbound flowspec NLRI.

Example:

```
Router(config)# router bgp 1
Router(config-bgp)# nsr
Router(config-bgp)# bgp router-id 10.2.3.4
Router(config-bgp)# address-family ipv4 flowspec
Router(config-bgp-af)# exit
Router(config-bgp)# address-family ipv6 flowspec
Router(config-bgp-af)# exit
Router(config-bgp)# address-family vpnv4 flowspec
Router(config-bgp-af)# exit
Router(config-bgp)# address-family vpnv6 flowspec
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.2.3.3
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# address-family ipv4 flowspec
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# address-family ipv6 flowspec
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# address-family vpnv4 flowspec
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# address-family vpnv6 flowspec
Router(config-bgp-nbr-af)# route-policy pass-all in
```

```
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# update-source Loopback0
```

Step 3 Define IPv4 traffic classes and associate them with a policy-map to specify actions.

Example:

```
Router(config)# class-map type traffic match-all ipv4_fragment
Router(config-cmap)# match destination-address ipv4 10.2.1.1 255.255.255.255
Router(config-cmap)# match source-address ipv4 172.16.0.1 255.255.255.255
Router(config-cmap)# end-class-map
Router(config)# class-map type traffic match-all ipv4_icmp
Router(config-cmap)# match destination-address ipv4 10.2.1.1 255.255.255.255
Router(config-cmap)# match source-address ipv4 172.16.0.1 255.255.255.255
Router(config-cmap)# end-class-map

Router(config)# policy-map type pbr scale_ipv4
Router(config-pmap)# class type traffic ipv4_fragment
Router(config-pmap-c)# drop
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic ipv4_icmp
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic class-default
Router(config-pmap-c)# end-policy-map
Router(config)# flowspec
Router(config)# address-family ipv4
Router(config-af)# service-policy type pbr scale_ipv4
Router(config-af)# exit
Router(config)# vrf vpn1
Router(config-vrf)# address-family ipv4
Router(config-vrf-af)# service-policy type pbr scale_ipv4
Router(config-vrf-af)# exit
Router(config-vrf)# exit
```

Step 4 Define IPv6 traffic classes and associate them with a policy-map for service policies.

Example:

```
Router(config)# flowspec
Router(config)# address-family ipv6
Router(config-af)# service-policy type pbr scale_ipv6
Router(config-af)# exit
Router(config)# vrf vpn1
Router(config-vrf)# address-family ipv6
Router(config-vrf-af)# service-policy type pbr scale_ipv6
Router(config-vrf-af)# exit
Router(config-vrf)# exit

Router(config)# class-map type traffic match-all ipv6_tcp
Router(config-cmap)# match destination-address ipv6 70:1:1::5a/128
Router(config-cmap)# match source-address ipv4 ipv6 80:1:1::5a/128
Router(config-cmap)# match destination-port 22
Router(config-cmap)# match source-port 4000
Router(config-cmap)# end-class-map
Router(config)# class-map type traffic match-all ipv6_icmp
Router(config-cmap)# match destination-address ipv6 70:2:1::1/128
Router(config-cmap)# match source-address ipv4 ipv6 80:2:1::1/128
Router(config-cmap)# end-class-map

Router(config)# policy-map type pbr scale_ipv6
Router(config-pmap)# class type traffic ipv6_tcp
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic ipv6_icmp
Router(config-pmap-c)# exit
```

```
Router(config-pmap)# class type traffic class-default
Router(config-pmap-c)# end-policy-map
Router(config)# flowspec
Router(config)# address-family ipv6
Router(config-af)# service-policy type pbr scale_ipv6
```

Step 5 Define a class map to match traffic based on DSCP values and a policy map to redirect IPv4 traffic to a next-hop with a specific police rate.

Example:

```
Router(config)# class-map type traffic match-all class_dscp_5
Router(config-cmap)# match destination-address ipv4 192.0.2.254 255.255.255.0
Router(config-cmap)# match dscp 10-12

Router(config-pmap)# class type traffic class_dscp_5
Router(config-pmap-c)# redirect ipv4 nexthop 10.26.245.2
Router(config-pmap-c)# police rate 5 mbps
Router(config-pmap-c)# root
```

This configuration establishes the router as a BGP flowspec server, allowing it to fully exchange flowspec rules for various traffic types with its client (10.2.3.3). It then defines specific traffic patterns for IPv4 and IPv6, applying actions such as dropping, redirecting, or rate-limiting these flows based on received rules, both globally and within a VPN.

Verify running configuration for BGP flowspec

The purpose of this running configuration is to verify a functional BGP flowspec client-server deployment. This setup enables a centralized server (Controller) to dynamically define, propagate, and enforce granular traffic filtering and policing rules across a client router, thereby providing a robust mechanism for network-wide traffic management and DDoS mitigation.

This configuration details the setup for two routers: a BGP Flowspec client (router ID 10.2.3.3) and a BGP Flowspec server (router ID 10.2.3.4), both within AS 1.

Procedure

Running configuration.

Example:

```
/* Define a Virtual Routing and Forwarding (VRF) instance named vrf1 and set up
import and export route targets for different address families. */
router bgp 140
 vrf vrf1
  address-family ipv4 unicast
   import route-target
    101:2000
    201:2000
   export route-target
    101:2000
    201:2000
   !
   address-family ipv4 flowspec
    import route-target
```

```
    101:2000
    201:2000
    export route-target
    101:2000
    201:2000
   !
   address-family ipv6 flowspec
    import route-target
    101:2000
    201:2000
    export route-target
    101:2000
    201:2000
   !
    address-family vpnv4 flowspec
    import route-target
    101:2000
    201:2000
    export route-target
    101:2000
    201:2000
   !
    address-family vpnv6 flowspec
     import route-target
     101:2000
     201:2000
     export route-target
     101:2000
     201:2000
    !

/* Configure BGP Flowspec both on the server and client side. */
flowspec
 address-family ipv4
  local-install interface-all
  !
  address-family ipv6
  local-install interface-all
  !
  address-family vpnv4
   local-install interface-all
  !
  address-family vpnv6
  local-install interface-all
  !

/* Configure the policy to accept all presented routes without modifying the routes. */
route-policy pass-all
 pass
 end-policy

/* Configure the policy to reject all presented routes without modifying the routes */
 route-policy drop-all
  drop
  end-policy

/* Configure BGP towards flowspec server */
router bgp 1
 nsr
  bgp router-id 10.2.3.3
   address-family ipv4 flowspec
   !
   address-family ipv6 flowspec
   !
```

```
   address-family vpnv4 flowspec
   !
   address-family vpnv6 flowspec
   !
    neighbor 10.2.3.4
     remote-as 1
      address-family ipv4 flowspec
       route-policy pass-all in
       route-policy drop-all out
       !
      address-family ipv6 flowspec
       route-policy pass-all in
       route-policy drop-all out
       !
      address-family vpnv4 flowspec
       route-policy pass-all in
       route-policy drop-all out
       !
       address-family vpnv6 flowspec
       route-policy pass-all in
       route-policy drop-all out
       !
       update-source Loopback0

/* Disable BGP Flowspec */
interface bundle-ether 3.1
 ipv4 flowspec disable
 ipv6 flowspec disable
/* The following section describes how you can configure BGP Flowspec on the server: */
/* Configure the policy to accept all presented routes without modifying the routes. */

route-policy pass-all
 pass
 end-policy

/* Configure the policy to reject all presented routes without modifying the routes */

route-policy drop-all
 drop
end-policy

/* Configure BGP towards flowspec client */
router bgp 1
 nsr
  bgp router-id 10.2.3.4
   address-family ipv4 flowspec
   !
   address-family ipv6 flowspec
   !
   address-family vpnv4 flowspec
   !
   address-family vpnv6 flowspec
   !
   neighbor 10.2.3.3
    remote-as 1
     address-family ipv4 flowspec
      route-policy pass-all in
      route-policy pass-all out
      !
     address-family ipv6 flowspec
      route-policy pass-all in
      route-policy pass-all out
        !
        address-family vpnv4 flowspec
```

```
         route-policy pass-all in
         route-policy pass-all out
        !
        address-family vpnv6 flowspec
         route-policy pass-all in
         route-policy pass-all out
         !
        update-source Loopback0


/* Configure IPv4 flowspec to be advertised to client. Define traffic classes. */
class-map type traffic match-all ipv4_fragment
 match destination-address ipv4 10.2.1.1 255.255.255.255
 match source-address ipv4 172.16.0.1 255.255.255.255end-class-map

class-map type traffic match-all ipv4_icmp
 match destination-address ipv4 10.2.1.1 255.255.255.255
 match source-address ipv4 172.16.0.1 255.255.255.255
 end-class-map

/* Define a policy map and associate it with traffic classes. */
policy-map type pbr scale_ipv4
 class type traffic ipv4_fragment
  drop
  !
  class type traffic ipv4_icmp
  !
  class type traffic class-default
  end-policy-map

flowspec
 address-family ipv4
  service-policy type pbr scale_ipv4
  !
  vrf vpn1
   address-family ipv4
    service-policy type pbr scale_ipv4
     !
      !

flowspec
 address-family ipv6
  service-policy type pbr scale_ipv6
  !
   vrf vpn1
    address-family ipv6
     service-policy type pbr scale_ipv6
      !
       !

/* Configure IPv6 flowspec to be advertised to client. Define traffic classes. */
class-map type traffic match-all ipv6_tcp
    match destination-address ipv6 70:1:1::5a/128
    match source-address ipv4 ipv6 80:1:1::5a/128
    match destination-port 22
    match source-port 4000
    end-class-map

class-map type traffic match-all ipv6_icmp
    match destination-address ipv6 70:2:1::1/128
    match source-address ipv4 ipv6 80:2:1::1/128
    end-class-map

/* Define a policy map and associate it with traffic classes. */
```

```
policy-map type pbr scale_ipv6
 class type traffic ipv6_tcp
  !
 class type traffic ipv6_icmp
  !
 class type traffic class-default
   end-policy-map

/* Class map configuration with DSCP. */
flowspec
 address-family ipv6
  service-policy type pbr scale_ipv6
class-map type traffic match-all class_dscp_5
 match destination-address ipv4 192.0.2.254 255.255.255.0
 match dscp 10-12

/* Policy map configuration with IPv4 Redirect and Rate Limiter */
 class type traffic class_dscp_5
  redirect ipv4 nexthop 10.26.245.2
   police rate 5 mbps
     root
```

Verify flowspec flow information, statistics, and NLRI data, and related policy maps

Verify the information about BGP Flowspec routes, including their attributes, status, and operational metrics. These outputs help users monitor and verify the distribution and application of Flowspec rules within the BGP routing environment, facilitating effective traffic filtering and mitigation of network threats.

Procedure

Step 1 Use the **show flowspec ipv4 detail** command to verify the Flowspec flow information and traffic statistics for IPv4.

Example:

```
Router# show flowspec ipv4 detail
Thu Jan 25 09:10:14.965 UTC

AFI: IPv4
  Flow           :Dest:10.0.0.1/8
    Actions      :Traffic-rate: 5000000 bps Redirect: VRF vpn1 Route-target: ASN2-1:1  (bgp.1)
    Statistics                       (packets/bytes)
      Matched             :                 200/25600
      Transmitted         :                 200/25600
      Dropped             :                   0/0
```

Step 2 Use the **show flowspec ipv6 detail** command to verify the Flowspec flow information and traffic statistics for IPv6.

Example:

```
Router# show flowspec ipv6 detail
AFI: IPv6
Flow:Dest:70:1:1::1/0-128,Source:80:1:1::1/0-128,NH:=6,DPort:=22,SPort:=4000,TCPFlags:=0x10,Length:=300,DSCP:=12

 Actions :Traffic-rate: 1000000 bps DSCP: cs1 Nexthop: 202:158:2::1 (bgp.1)
 Statistics (packets/bytes)
```

```
    Matched         : 64091597/19483845488
    Transmitted     : 33973978/10328089312
    Dropped         : 30117619/9155756176
```

Step 3 Use the **show flowspec vrf customer_1 ipv4 detail** command to verify the Flowspec flow information and statistics for IPv4 within the specified VRF.

Example:

```
show flowspec vrf customer_1 ipv4 detail
VRF: customer_1 AFI: IPv4
Flow :Dest:202.158.3.2/32,Source:202.158.1.2/32
Actions :Traffic-rate: 250000000 bps DSCP: cs6 Redirect: VRF dirty_dancing
Route-target: ASN2-4787:666 (bgp.1)
Statistics (packets/bytes)
Matched : 37260786850/4098686553500
Transmitted : 21304093027/2343450232970
Dropped : 15956693823/1755236320530
```

Step 4 Use the **show flowspec vrf customer_1 ipv6 detail** command to verify the Flowspec flow information and statistics for IPv6 within the specified VRF.

Example:

```
Router# show flowspec vrf customer_1 ipv6 detail
VRF: customer_1 AFI: IPv6
Flow
:Dest:200:158:3::2/0-128,Source:200:158:1::2/0-128,NH:=6,DPort:=22,SPort:=4000,Length:=300,DSCP:=12
Actions :Traffic-rate: 250000000 bps DSCP: cs6 Redirect: VRF dirty_dancing
Route-target: ASN2-4787:666 (bgp.1)
Statistics (packets/bytes)
Matched : 16130480136/4903665961344
Transmitted : 8490755776/2581189755904
Dropped : 7639724360/2322476205440
```

Step 5 Use the **show flowspec ipv4 nlri** command to verify the NLRI in hex format for IPv4 Flowspec routes.

Example:

```
Router# show flowspec ipv4 nlri
AFI: IPv4
NLRI (hex) :0x0120460101010381060581500691 0bb80a81c80b810a
Actions :Traffic-rate: 0 bps (bgp.1)
```

Step 6 Use the **show flowspec ipv6 nlri** command to verify the NLRI in hex format for IPv6 Flowspec routes.

Example:

```
Router# show flowspec ipv6 nlri
AFI: IPv6
NLRI (hex)
:0x018000007000010001000000000000000000010280000080000100010000000000000000000010381060581160691 0fa00981100a91012c0b810c
Actions :Traffic-rate: 1000000 bps DSCP: cs1 Nexthop: 202:158:2::1 (bgp.1)
```

Step 7 Use the **show flowspec vrf customer_1 ipv4 nlri** command to verify the NLRI in hex format for IPv4 Flowspec routes within the specified VRF.

Example:

```
Router# show flowspec vrf customer_1 ipv4 nlri
VRF: customer_1 AFI: IPv4
NLRI (hex) :0x0120ca9e03020220ca9e0102
Actions :Traffic-rate: 250000000 bps DSCP: cs6 Redirect: VRF dirty_dancing
Route-target: ASN2-4787:666 (bgp.1)
```

Step 8 Use the **show flowspec vrf customer_1 ipv6 nlri** command to verify the NLRI in hex format for IPv6 Flowspec routes within the specified VRF.

Example:

```
Router# show flowspec vrf customer_1 ipv6 nlri
VRF: customer_1 AFI: IPv6
NLRI (hex)
:0x0180000200015800030000000000000000000202800002000158000100000000000000000002038106058116069 10fa00a91012c0b810c
Actions :Traffic-rate: 250000000 bps DSCP: cs6 Redirect: VRF dirty_dancing
Route-target: ASN2-4787:666 (bgp.1)
```

Step 9 Use the **show policy-map transient type pbr** command to verify the policy-map dynamically created for Flowspec-based traffic policies.

Example:

```
Router# show policy-map transient type pbr
policy-map type pbr __bgpfs_default_IPv4
handle:0x36000004
table description: L3 IPv4 and IPv6
class handle:0x760013eb sequence 1024
match destination-address ipv4 10.1.1.1 255.255.255.255
match protocol tcp
match destination-port 80
match source-port 3000
```

The show outputs provides the user with the visibility into the Flowspec environment, facilitating validation and maintenance of security policies within the BGP routing framework.

Verify the BGP flowspec on the client

Verify the correct reception, installation, and propagation of BGP Flowspec routes for IPv4, IPv6, VPNv4, and VPNv6 address families. Confirm that Flowspec information exchanges properly with BGP peers.

Before you begin

Ensure that you have appropriate user privileges to execute privileged EXEC mode commands.

Follow these steps to verify the BGP flowspec on the client.

Procedure

Step 1 Use the **show bgp ipv4 flowspec** command to verify BGP Flowspec routes for the IPv4 address family.

Example:

```
Router# show bgp ipv4 flowspec
GP router identifier 202.158.0.1, local AS number 4787
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0 RD version: 7506
BGP main routing table version 7506
BGP NSR Initial initsync version 130 (Reached)
BGP NSR/ISSU Sync-Group versions 7506/0
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
*>iDest:10.1.1.1/32,Proto:=6,DPort:=80,SPort:=3000,Length:=200,DSCP:=10/176
0.0.0.0 10 0 ?
*>iDest:10.1.1.2/32,Proto:=6,DPort:=80,SPort:=3000,Length:=200,DSCP:=10/176
0.0.0.0 10 0 ?
*>iDest:10.1.1.3/32,Proto:=6,DPort:=80,SPort:=3000,Length:=200,DSCP:=10/176
0.0.0.0 10 0 ?
*>iDest:10.1.1.4/32,Proto:=6,DPort:=80,SPort:=3000,Length:=200,DSCP:=10/176
0.0.0.0 10 0 ?
*>iDest:10.1.1.5/32,Proto:=6,DPort:=80,SPort:=3000,Length:=200,DSCP:=10/176
0.0.0.0 10 0 ?
```

Step 2 Use the **show bgp ipv6 flowspec** command to verify BGP Flowspec routes for the IPv6 address family.

Example:

```
Router# show bgp ipv6 flowspec
BGP router identifier 202.158.0.1, local AS number 4787
...
Network Next Hop Metric LocPrf Weight Path
*>iDest:70:1:1::1/0-128,Source:80:1:1::1/0-128,NH:=6,DPort:=22,SPort:=4000,TCPFlags:=0x10,Length:=300,DSCP:=12/464
202:158:2::1 100 0 i
*>iDest:70:1:1::2/0-128,Source:80:1:1::2/0-128,NH:=6,DPort:=22,SPort:=4000,TCPFlags:=0x10,Length:=300,DSCP:=12/464
202:158:2::1 100 0 i
*>iDest:70:1:1::3/0-128,Source:80:1:1::3/0-128,NH:=6,DPort:=22,SPort:=4000,TCPFlags:=0x10,Length:=300,DSCP:=12/464
202:158:2::1 100 0 i
*>iDest:70:1:1::4/0-128,Source:80:1:1::4/0-128,NH:=6,DPort:=22,SPort:=4000,TCPFlags:=0x10,Length:=300,DSCP:=12/464
202:158:2::1 100 0 i
*>iDest:70:1:1::5/0-128,Source:80:1:1::5/0-128,NH:=6,DPort:=22,SPort:=4000,TCPFlags:=0x10,Length:=300,DSCP:=12/464
202:158:2::1 100 0 i
```

Step 3 Use the **show bgp vpnv4 flowspec** command to verify BGP Flowspec routes for VPNv4 address families.

Example:

```
Router# show bgp vpnv4 flowspec
BGP router identifier 202.158.0.1, local AS number 4787
...
Route Distinguisher: 202.158.0.1:0 (default for vrf customer_1)
*>iDest:202.158.3.2/32,Source:202.158.1.2/32/96
0.0.0.0 100 0 i
Route Distinguisher: 202.158.0.2:1
*>iDest:202.158.3.2/32,Source:202.158.1.2/32/96
0.0.0.0 100 0 i
Processed 2 prefixes, 2 paths
```

Step 4 Use the **show bgp vpnv6 flowspec** command to verify BGP Flowspec routes for VPNv6 address families.

Example:

```
Router# show bgp vpnv6 flowspec

BGP router identifier 202.158.0.1, local AS number 4787
...
Route Distinguisher: 202.158.0.1:0 (default for vrf customer_1)
*>iDest:200:158:3::2/0-128,Source:200:158:1::2/0-128,NH:=6,DPort:=22,SPort:=4000,Length:=300,DSCP:=12/440
0.0.0.0 100 0 i
Route Distinguisher: 202.158.0.2:1
*>iDest:200:158:3::2/0-128,Source:200:158:1::2/0-128,NH:=6,DPort:=22,SPort:=4000,Length:=300,DSCP:=12/440
0.0.0.0 100 0 i
Processed 2 prefixes, 2 paths
```

Step 5 Use the **show bgp ipv6 flowspec summary** command to view a summary of BGP Flowspec information for IPv6.

Example:

```
Router# show bgp ipv6 flowspec summary
BGP router identifier 202.158.0.1, local AS number 4787
...
Neighbor Spk AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down St/PfxRcd
202.158.2.1 0 4787 1548 1648 1504 0 0 1d01h 750 <-- this
many flowspecs were received from server
202.158.3.1 0 4787 1683 1644 1504 0 0 1d01h 751
202.158.4.1 0 4787 1543 1649 1504 0 0 1d01h 0
```

Step 6 Use the **show bgp vpnv4 flowspec summary** command to view a summary of BGP Flowspec information for VPNv4.

Example:

```
Router# show bgp vpnv4 flowspec summary

BGP router identifier 202.158.0.1, local AS number 4787
...
Neighbor Spk AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down St/PfxRcd
02.158.2.1 0 4787 1549 1648 5 0 0 1d01h 1 <-- this
many flowspecs were received from server
202.158.3.1 0 4787 1684 1644 5 0 0 1d01h 0
202.158.4.1 0 4787 1543 1649 5 0 0 1d01h 0
```

Step 7 Use the **show bgp vpnv6 flowspec summary** command to view a summary of BGP Flowspec information for VPNv6.

Example:

```
Router# show bgp vpnv6 flowspec summary
BGP router identifier 202.158.0.1, local AS number 4787
...
Neighbor Spk AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down St/PfxRcd
202.158.2.1 0 4787 1549 1649 5 0 0 1d01h 1 <-- this
many flowspecs were received from server
202.158.3.1 0 4787 1684 1645 5 0 0 1d01h 0
202.158.4.1 0 4787 1543 1650 5 0 0 1d01h 0
```

Step 8 Use the **show flowspec vrf all afi-all summary** command to view a summary of Flowspec routes and service policies across all VRFs and address families.

Example:

```
Router# show flowspec vrf all afi-all summary
Flowspec VRF+AFI table summary:
VRF: default
  AFI: IPv4
    Total Flows:              1
    Total Service Policies:   1
VRF: default
  AFI: IPv6
    Total Flows:              0
    Total Service Policies:   0
```

The show outputs confirm that BGP Flowspec routes for IPv4, IPv6, VPNv4, and VPNv6 address families are present and match the expected destinations and policies. Additionally, the summary commands show that Flowspec routes are being successfully received from specific BGP neighbors, verifying correct propagation and peering.

BGP flowspec redirect from global VRF to L3VPN and segment routing policy

The BGP flowspec redirect from Global VRF to L3VPN and Segment Routing policy is a BGP routting feature that

- dynamically redirect traffic to the VRF table
- enable traffic to search for the destination IP address within the L3VPN or via a segment routing policy, and
- improve routing adaptability and service continuity.

This feature also allows you to execute precise traffic actions, which optimize network performance and security.

When traffic arrives with a destination IP address not found in the global routing table but present in a VRF routing table, this feature ensure the packet is redirected correctly to the customer VRF. This redirection enhances routing flexibility and maintains service continuity by leveraging L3VPN or segment routing policies for forwarding decisions.

This capability helps you manage complex network environments by applying fine-grained traffic control and improving overall network efficiency.

Table 49: Feature History Table

Feature Name	Release Name	Description
BGP flowspec redirect from global VRF to L3VPN and segment routing policy	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
BGP flowspec redirect from global VRF to L3VPN and segment routing policy	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Name	Description
BGP flowspec redirect from global VRF to L3VPN and segment routing policy	Release 24.2.11	You can now enhance network routing efficiency by enabling BGP Flowspec to dynamically redirect traffic to the VRF table, where the traffic searches for the destination IP address either within the L3VPN or via a segment routing policy. This improvement boosts routing adaptability and service continuity. Additionally, the protocol extension equips you to execute precise traffic actions, optimizing network performance and security.

How BGP flowspec redirect from global VRF to L3VPN and segment routing policy works

This process redirects packets arriving on a global VRF interface with destination IPs found only in a customer VRF. Without this redirect, packets are dropped. The BGP Flowspec server programs and sends redirect rules via BGP NLRI to neighbors, which store and activate them. Matching packets are then forwarded to the correct VRF using the specified route target through L3VPN or Segment Routing Policy. This ensures accurate routing and precise traffic control in complex networks.

Summary

The key components involved in this process are:

- BGP Flowspec server: Programs and distributes redirect rules using BGP NLRI to neighbors.
- BGP Flowspec neighbors (clients): Store and activate the redirect rules in their databases.
- Network interfaces (for example, VRFA): Receive incoming packets that require VRF-specific routing.
- Routing tables (global VRF and customer VRF): Contain destination IP information used for packet forwarding.
- L3VPN and Segment Routing Policy (SR-Policy): Mechanisms used to forward packets to the correct VRF instance.

Workflow

Figure 27: Forwarding based on SR-Policy

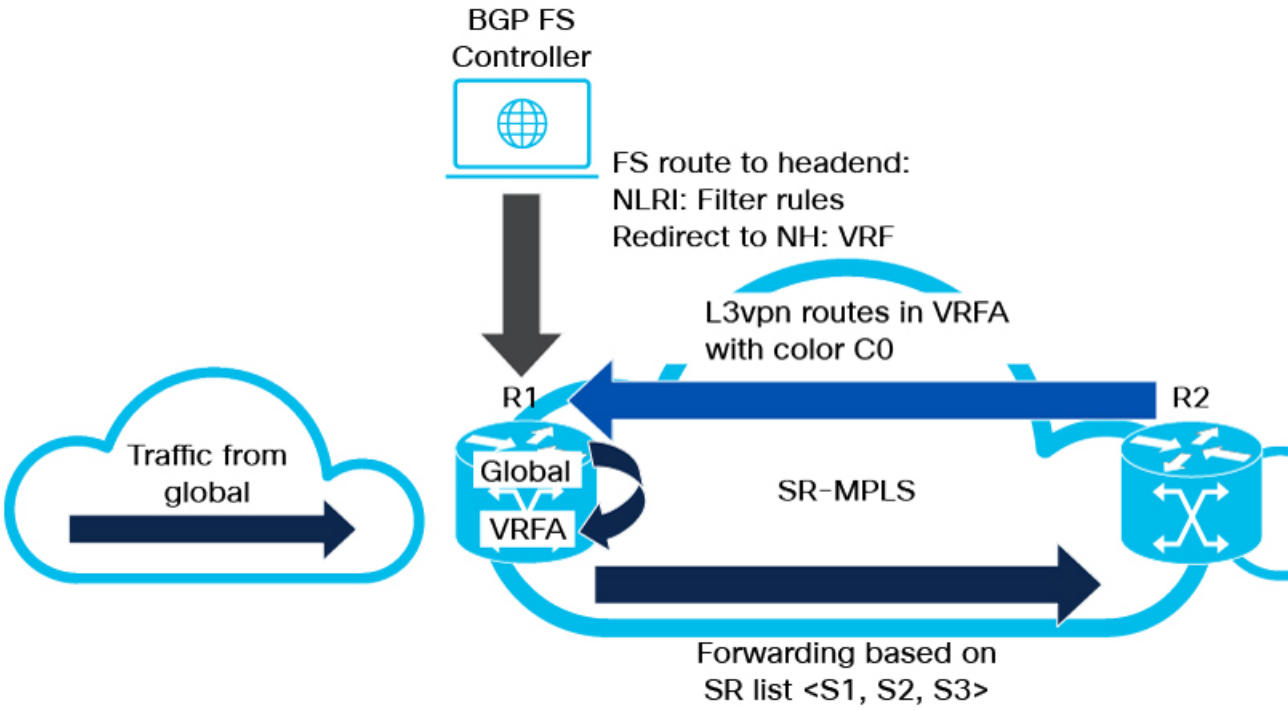
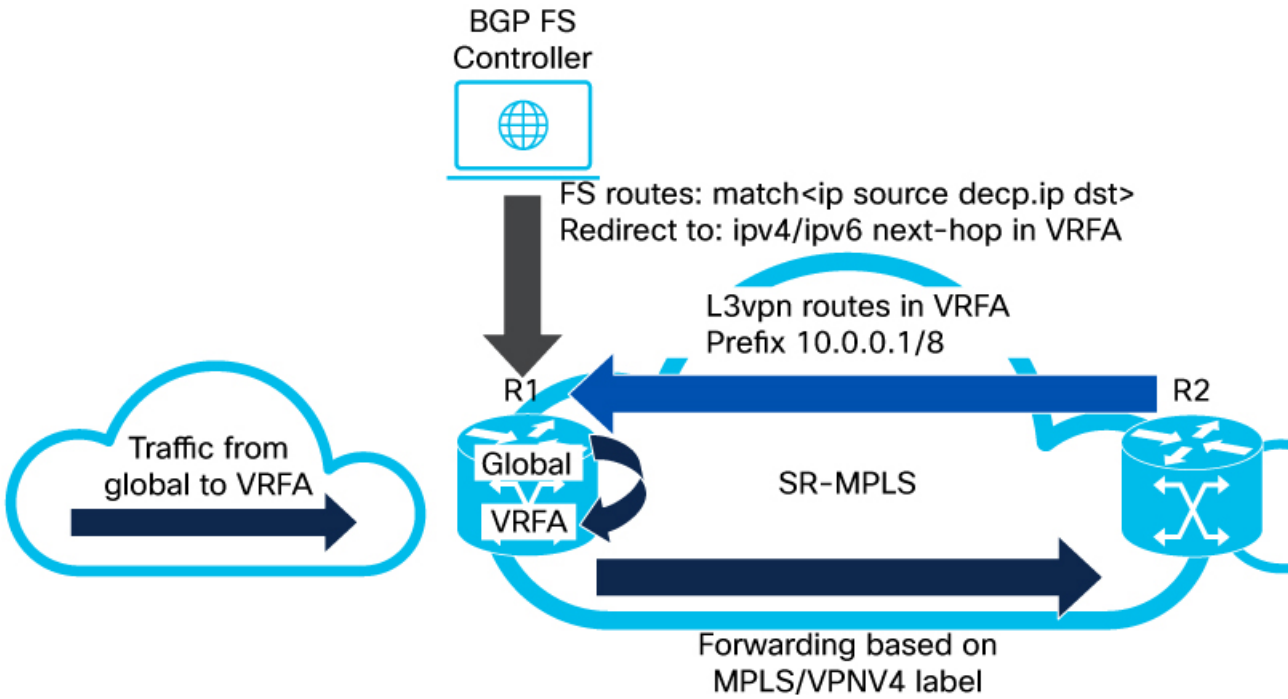


Figure 28: Forwarding based on MPLS



These stages describe How BGP flowspec redirect from global VRF to L3VPN and segment routing policy works.

1. Incoming packets arrive on a global VRF interface with destination IPs that exist only in a customer VRF routing table.
2. Without intervention, these packets are dropped due to lookup failure in the global VRF.
3. The BGP Flowspec server programs redirect rules and propagates them to BGP Flowspec neighbors via BGP NLRI.
4. Neighbors store and activate these rules in their databases.
5. Incoming packets matching the active rules are redirected to the appropriate customer VRF instance.
6. The redirect action specifies the correct route target to ensure accurate routing.
7. Forwarding occurs through L3VPN or Segment Routing Policy, enabling precise traffic control and preventing packet loss.

Result

This process ensures that packets with destination IPs only present in customer VRFs are correctly redirected and forwarded, preventing packet drops and enabling fine-grained traffic control within complex network environments.

Configure BGP flowspec redirect from global VRF

Enable BGP Flowspec redirect from the global VRF to customer VRFs by configuring class maps, policy maps, and applying service policies on the BGP Flowspec controller.

This task enables BGP Flowspec redirect from the global VRF to customer VRFs to prevent packet drops when destination IPs exist only in customer VRF routing tables. The BGP Flowspec server programs and distributes redirect rules to neighbors, which store and activate them. Matching packets are redirected to the correct VRF via the specified route target, ensuring accurate forwarding through L3VPN or Segment Routing Policy for precise traffic control in complex networks.

Before you begin

- Ensure BGP Flowspec is supported and enabled on the router.
- Confirm that the global VRF and customer VRFs are properly configured.
- Verify that L3VPN or Segment Routing Policy (SR-Policy) mechanisms are in place for forwarding.
- Have access to configure class maps, policy maps, and flowspec service policies on the BGP Flowspec controller.
- Confirm that route targets for the customer VRFs are defined and reachable.
- Ensure a policer action is configured to enable BGP Flowspec statistics.

Follow these steps to configure BGP flowspec redirect from global VRF.

Procedure

Step 1 Define traffic classes to match packets based on destination IP addresses.

Example:

```
Router# configure terminal
Router(config)# class-map type traffic match-all ipv4_CM1
Router(config-cmap)# match destination-address ipv4 10.0.0.1 255.255.255.0
Router(config-cmap)# end-class-map
Router(config)# class-map type traffic match-all ipv6_CM1
Router(config-cmap)# match destination-address ipv6 2000:0:0:1::/64
Router(config-cmap)# end-class-map
Router(config)# exit
```

Step 2 Create policy maps to specify redirect actions for the matched traffic classes.

Example:

```
Router(config)# policy-map type pbr ipv4_PM1
Router(config-pmap)# class type traffic ipv4_CM1
Router(config-pmap-c)# redirect nexthop route-target 1:1
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic class-default
Router(config-pmap)# end-policy-map
Router(config)# policy-map type pbr ipv6_PM1
Router(config-pmap)# class type traffic ipv6_CM1
Router(config-pmap-c)# redirect nexthop route-target 1:1
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic class-default
Router(config-pmap)# end-policy-map
Router(config)# exit
```

Step 3 Attach the policy maps to the respective address families under flowspec configuration.

Example:

```
Router(config)# flowspec
Router(config)# address-family ipv4
Router(config-af)# service-policy type pbr ipv4_PM1
Router(config-af)# exit
Router(config)# address-family ipv6
Router(config-af)# service-policy type pbr ipv6_PM1
Router(config-af)# exit
Router(config)# exit
```

Step 4 Install flowspec rules on all interfaces locally:

Example:

```
Router(config)# flowspec
Router(config)# local-install interface-all
Router(config)# exit
```

Step 5 Verify the running configuration.

Example:

```
Router# show running-config
class-map type traffic match-all ipv4_CM1
 match destination-address ipv4 10.0.0.1. 255.255.255.0
 end-class-map
!
```

```
class-map type traffic match-all ipv6_CM1
 match destination-address ipv6 2000:0:0:1::/64
 end-class-map
!

policy-map type pbr ipv4_PM1
 class type traffic ipv4_CM1
  redirect nexthop route-target 1:1

  !
 !
 class type traffic class-default
 !
 end-policy-map
!
policy-map type pbr ipv6_PM1
 class type traffic ipv6_CM1
  redirect nexthop route-target 1:1

  !
 !
 class type traffic class-default
 !
 end-policy-map
!

flowspec
 address-family ipv4
  service-policy type pbr ipv4_PM1
 address-family ipv6
  service-policy type pbr ipv6_PM1


flowspec config on PE1:

flowspec
 local-install interface-all
```

Step 6 Use the **show ofa objects pbr object-count location 0/RP0/CPU0** command in privileged EXEC mode to verify the number of BGP Flowspec entries in the OFA object.

Example:

```
Router# show ofa objects pbr object-count location 0/RP0/CPU0

Table [PBR] has 4200 entries in DB
Table [PBR] had 4200 as highest count @ Tue Feb  6 20:08:04 2024
```

You will see the current count of BGP Flowspec entries in the OFA object and detailed statistics for each BGP Flowspec rule, including matched, transmitted, and dropped packets and bytes.

Step 7 Use the **show flowspec ipv4 detail** command in privileged EXEC mode to verify the BGP Flowspec rules and their statistics.

Example:

```
Router# show flowspec ipv4 detail
AFI: IPv4
  Flow           :Dest:10.0.0.1/8
    Actions      :Traffic-rate: 5000000 bps Redirect: VRF vpn1 Route-target: ASN2-1:1  (bgp.1)
    Statistics                       (packets/bytes)
      Matched             :                 200/25600
      Transmitted         :                 200/25600
      Dropped             :                   0/0
  Flow           :Dest:10.0.0.2/8
```

```
Actions      :Traffic-rate: 5000000 bps Redirect: VRF vpn1 Route-target: ASN2-1:1  (bgp.1)
Statistics                          (packets/bytes)
  Matched             :                 200/25600
  Transmitted         :                 200/25600
  Dropped             :                   0/0
```

Traffic filtering actions: what you need to know about controlling traffic with BGP flowspec

You use traffic filtering actions to control how IP traffic matching specific flow rules is handled. This topic explains how to specify actions like dropping or policing traffic using BGP flowspec extended communities. Understanding these actions helps you manage network traffic effectively and maintain performance and security.

The default action accepts IP traffic that matches a flow specification rule.

You can change this behavior by applying extended community values that specify different actions.

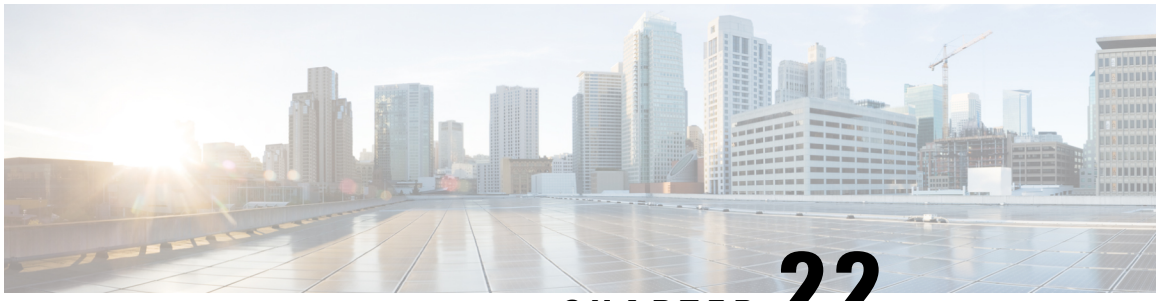
Table 50: Traffic Filtering Actions

Type	Extended Community	PBR Action	Description
0x8006	traffic-rate 0 traffic-rate <rate>	Drop Police	The traffic-rate extended community is a non-transitive extended community across the autonomous-system boundary and uses following extended community encoding: The first two octets carry the 2-octet id, which can be assigned from a 2-byte AS number. When a 4-byte AS number is locally present, the 2 least significant bytes of such an AS number can be used. This value is informational. The remaining 4 octets carry the rate information in IEEE floating point [IEEE.754.1985] format, bytes per second. A traffic-rate of 0 should result on all traffic for the particular flow to be discarded. Command syntax **police rate < > | drop**
0x8009	traffic-marking	Set DSCP	The traffic marking extended community instructs a system to modify the differentiated service code point (DSCP) bits of a transiting IP packet to the corresponding value. This extended community is encoded as a sequence of 5 zero bytes followed by the DSCP value encoded in the 6 least significant bits of 6th byte. Command syntax **set dscp** *<6 bit value>*

Type	Extended Community	PBR Action	Description
0x0800	Redirect IP NH	Redirect IPv4 or IPv6 Nexthop	Announces the reachability of one or more flowspec NLRI. When a BGP speaker receives an UPDATE message with the redirect-to- IP extended community it is expected to create a traffic filtering rule for every flow-spec NLRI in the message that has this path as its best path. The filter entry matches the IP packets described in the NLRI field and redirects them or copies them towards the IPv4 or IPv6 address specified in the Network Address of Next-Hop field of the associated MP_REACH_NLRI. **Note** The redirect-to-IP extended community is valid with any other set of flow-spec extended communities except if that set includes a redirect-to-VRF extended community (type 0x8008) and in that case the redirect-to-IP extended community should be ignored. **Note** Redirect IP NH is supported only in default VRF. Command syntax **redirect {ipv4 | ipv6} next-hop {ipv4-address | ipv6-address}**

Note

- You cannot use the BGP flowspec actions *rate limit* and *redirect*together.
- The *redirect* action works only with nexthop IPv4 and IPv6 addresses, not with nexthop VRF IPv4 or IPv6.



CHAPTER 22

BGP Session Security Mechanisms

This chapter provides an overview of essential security mechanisms for protecting BGP sessions on Cisco routers. It covers key features such as BGP keychains, Martian address checks, TTL security (GTSM), interface-based LPTS identifiers, and prefix origin validation using RPKI. Each section explains the purpose of the mechanism and offers practical configuration guidance to help secure BGP routing against common threats.

BGP keychains

A BGP keychain is a security mechanism that

- enables keychain authentication between two BGP peers based on standardized protocols
- allows hitless key rollover for authentication using time-based specifications, and
- provides a configurable tolerance window to handle clock skew between endpoints for seamless operation.

Keychain interoperability and behavior

Both BGP endpoints must comply with the draft-bonica-tcp-auth-05.txt standard for keychain authentication to function. A keychain on one endpoint and a password on the other will not work. The configurable tolerance window extends the accept period to allow for clock differences and maintains hitless key rollover for applications such as routing and management protocols.

If there is a keychain configuration mismatch at the endpoints resulting in no common keys, BGP session traffic (send or accept) may be interrupted. Otherwise, the key rollover does not disrupt the BGP session.

Configure keychains for BGP

Configure BGP keychains to secure authentication for BGP sessions using MAC authentication algorithms and enable graceful key rollover.

BGP keychains enhance the security of BGP routing by providing flexible authentication options and key management. This is especially useful in environments where multiple neighbors or session groups need secure, easily managed authentication.

Before you begin

- Ensure you have a defined keychain with the necessary keys and authentication parameters.
- Identify the autonomous system (AS) numbers for your router and remote neighbors.

Procedure

Enter BGP configuration mode, and configure keychain-based authentication for the neighbor.

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# neighbor 172.16.40.24
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# keychain kych_a
```

Note
If a keychain is configured for a neighbor group or session group, a neighbor using the group inherits the keychain. Values configured directly for a neighbor override any inherited values.

Martian address checks

A Martian address check is a router security feature that

- prevents routers from accepting packets with reserved or illogical IP address prefixes
- is applied by default in BGP configurations to drop packets originating from Martian addresses, and
- can be disabled to allow routers to process routes from specific sites using designated IPv4 or IPv6 prefixes.

Martian addresses are reserved or undefined IP address ranges that should not appear in legitimate internet routing tables. Filtering these addresses improves network security by helping ensure that only valid, routable addresses are accepted during routing.

Examples

Common Martian address prefixes include:

- IPv4:

- 0.0.0.0/8
- 127.0.0.0/8
- 224.0.0.0/4

- IPv6:
  - ::
  - ::0002 through ::ffff
  - ::ffff:a.b.c.d
  - fe80:xxxx
  - ffxx:xxxx

Restrictions:

Routers running OSPF or IS-IS protocols cannot access routes with Martian address prefixes, even if the Martian address check is disabled.

Disable the Martian address check in BGP

By default, Cisco routers drop routes and packets with Martian (reserved or unusual) IP prefixes during BGP operations. You may need to override this security check to allow routing for certain special network scenarios.

Before you begin

Make sure you have console or privileged EXEC access to the Cisco 8000 Series Router.

Procedure

Step 1 Enter router BGP configuration mode, and use the **default-martian-check disable** command to disable the Martian address check.

Example:

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# default-martian-check disable
Router(config-bgp-af)# commit
```

Step 2 Use the **show bgp ipv4 unicast** command or **show bgp ipv6 unicast** command to check whether the Martian address check is enabled or disabled in your BGP configuration.

Example:

```
Router# show bgp ipv6 unicast
BGP router identifier 10.2.2.1, local AS number 1
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800000 RD version: 29
BGP main routing table version 29
```

```
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
Dampening enabled
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
*>i::/0              1:1:1:1:1:1:1:1         100        0             i
* i192:1::/112       1.1.1.1                   0      100             0 ?
*>i                  1:1:1:1:1:1:1:1           0      100             0 ?
* iff11:1123::/64    1.1.1.1                   2      100             0 ?
*>i                  1:1:1:1:1:1:1:1           2      100             0 ?
```

BGP eBGP security GTSM

BGP eBGP security GTSM is a BGP security feature that

- restricts accepted IP packets to those with a Time to Live (TTL) or Hop Limit equal to the maximum value for eBGP neighbors
- protects a router's control plane from CPU-utilization attacks caused by forged protocol packets, and
- applies robust session security for eBGP peerings, especially between directly connected or loopback-adjacent routers.

Table 51: Feature history table

Feature Name	Release Information	Feature Description
BGP-eBGP Security GTSM	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
BGP-eBGP Security GTSM	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
BGP-eBGP Security GTSM	Release 7.3.1	The Generalized TTL Security Mechanism (GTSM) is designed to protect a router's IP-based control plane from CPU-utilization based attacks. This feature enables the router to accept only IP packets with a TTL count that is equal to the maximum TTL value. New command introduced: • **ttl-security**

Generalized TTL Security Mechanism (GTSM)

GTSM leverages the fact that most protocol peerings occur between adjacent routers or loopback addresses. Since TTL spoofing is nearly impossible in these scenarios, enforcing maximum TTL value acceptance creates a simple and effective defense against infrastructure attacks using forged packets. GTSM applies to both IPv4 (TTL) and IPv6 (Hop Limit) sessions.

How GTSM Works

When GTSM is enabled, the router only accepts packets whose TTL equals the maximum value. Packets with lower TTL are discarded and do not generate ICMP responses, preventing feedback to attackers.

Configure BGP eBGP security GTSM

Secure eBGP neighbor sessions using the Generalized TTL Security Mechanism (GTSM).

Before you begin

Identify the eBGP neighbor address and relevant autonomous system numbers.

Procedure

Step 1 Enter router BGP configuration mode, set the eBGP multihop value, and use the **ttl-security** command to enable GTSM for the eBGP neighbor.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 2001::db8
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# ebgp-multihop 255
Router(config-bgp-nbr)# ttl-security
Router(config-bgp-nbr)# address-family ipv6 unicast
Router(config-bgp-nbr-af)# multipath
Router(config-bgp-nbr-af)# route-policy PASS_ALL in
Router(config-bgp-nbr-af)# route-policy PASS_ALL out
```

Step 2 (Optional) Enable multipath for redundancy or load balancing, and apply route policies as required.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 2001::db8
```

```
Router(config-bgp-nbr)# address-family ipv6 unicast
Router(config-bgp-nbr-af)# multipath
Router(config-bgp-nbr-af)# route-policy PASS_ALL in
Router(config-bgp-nbr-af)# route-policy PASS_ALL out
```

Interface-based LPTS identifiers

An interface-based LPTS identifier is a network security feature that

- associates each directly connected external BGP (eBGP) neighbor with a specific router interface
- restricts inbound traffic so only packets originating from a designated eBGP neighbor can traverse through the mapped interface, and
- prevents IP spoofing and session hijacking attempts by enforcing strict interface-level packet filtering and policing.

Table 52: Feature History Table

Feature Name	Release Name	Description
Protection of Directly Connected EBGP Neighbors through Interface-Based LPTS Identifier	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Protection of Directly Connected EBGP Neighbors through Interface-Based LPTS Identifier	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Name	Description
Protection of Directly Connected EBGP Neighbors through Interface-Based LPTS Identifier	Release 7.10.1	We have enhanced the network security for directly connected eBGP neighbors by ensuring that only packets originating from designated eBGP neighbors can traverse through a single interface, thus preventing IP spoofing. This is made possible because we've now added an interface identifier for Local Packet Transport Services (LPTS). LPTS filters and polices the packets based on the type of flow rate you configure. The feature introduces these changes: **CLI:** • **bgp lpts-secure-binding** **YANG Data Model:** • `Cisco-IOS-XR-um-router-bgp-cfg` (see GitHub, YANG Data Models Navigator)

Overview of Local Packet Transport Services (LPTS) in BGP

LPTS maintains tables describing all packet flows destined for the secure domain router (SDR), ensuring packets are delivered only to their intended destinations. In BGP sessions, LPTS entries are categorized as follows:

- **BGP known:** Entries for established BGP neighbors.
- **BGP configured peer:** Entries for initial packets (TCP SYN and 3rd ACK) from specifically configured BGP neighbors.
- **BGP default entries:** Entries for all packets from unconfigured BGP neighbors.

Security enhancement with interface identifier

By adding an interface identifier to LPTS entries for directly connected eBGP neighbors, the router ensures that only traffic from the designated interface and neighbor IP can match the LPTS entry and reach the BGP session. Spoofed packets from other interfaces, even with correct IP/port/VRF combinations, only match the default LPTS entry where they are policed and forwarded to TCP for reset generation. This prevents attackers from exploiting established session entries by flooding from other interfaces.

Conditions for passing the interface identifier

The interface identifier is passed to LPTS and TCP only when all these conditions are met:

- The BGP peer is configured as external (eBGP).

- Fast External Failover (FEF) is not disabled.
- The BGP peer is directly connected.
- The BGP peer is not a dynamic peer.
- eBGP multihop is not enabled.
- Default eBGP TTL is used.
- The "ignore connected" option is not configured.
- A non-link local IPv6 neighbor address is configured.

Interface identifier binding during session establishment

During session establishment, both passive (received connections) and active (initiated connections) BGP bindings supply the interface identifier, so the LPTS entry for the connection is tightly bound to the specified interface.

Interface-based LPTS identification example

Suppose an attacker floods packets matching the established BGP session (source IP, destination IP, source port, destination port, VRF) from an unintended interface. With interface-based LPTS identification enabled, those packets do not match the LPTS entry for the legitimate peer; they are discarded or strictly policed, ensuring BGP session stability and preventing flapping.

Configure LPTS secure binding for directly connected EBGP neighbors

Enable secure binding between LPTS and directly connected eBGP neighbors to enhance network protection.

Procedure

Step 1 Enter BGP configuration mode, and enable LPTS secure binding for BGP.

Example:

```
Router#(config)router bgp 100
Router#(config-bgp) bgp lpts-secure-binding
```

Step 2 Confirm that LPTS secure binding is enabled.

Example:

```
Router# show bgp process | in LPTS

Wed Dec 14 14:28:33.779 PST
LPTS secure binding is enabled
```

Step 3 Verify that LPTS entries now include interface handle identifiers.

Example:

```
Router# show lpts pifib entry brief
```

```
IPv4    default  TCP    any          [0x00000003]      10.10.10.1,23756 10.10.10.2,179
IPv4    default  TCP    any          0/0/CPU0           10.10.10.1,179 10.10.10.2
IPv4    default  TCP    Gi0/2/0/1    [0x00000003]       192.0.2.1,57342 192.0.2.3,179
IPv4    default  TCP    Gi0/2/0/1    0/0/CPU0           192.0.2.1,179 192.0.2.3
IPv4    default  TCP    any          [0x00000003]       209.165.201.1,179 209.165.201.4,52798
IPv4    default  TCP    any          0/0/CPU0           209.165.201.1,179 209.165.201.0/24
IPv4    default  TCP    Gi0/2/0/3    [0x00000003]       172.16.0.1,179 172.16.0.5,49505
IPv4    default  TCP    Gi0/2/0/3    0/0/CPU0           172.16.0.1,179 172.16.0.5
IPv4    default  TCP    any          [0x00000003]       192.168.0.1,179 192.168.0.6,32909
IPv4    default  TCP    any          0/0/CPU0           192.168.0.1,179 192.168.0.6
```

Step 4 Verify that the status of the connected interface handle in LPTS is active for the eBGP neighbor.

Example:

```
Router# show bgp neighbor 192.0.2.3, detail | in Connected

Wed Dec 14 14:28:51.814 PST
  Connected IFH: 0x1000080, IFH in LPTS 0x1000080
```

BGP prefix origin validation mechanisms

A BGP prefix origin validation mechanism is a route security feature that

- uses the Resource Public Key Infrastructure (RPKI) to validate the Autonomous System (AS) originating a BGP prefix
- prevents prefix mis-announcement by verifying that the origin AS claiming an address prefix is authorized, and
- enhances routing security by ensuring that BGP routers accept only prefixes with verifiable, legitimate origin AS numbers.

Table 53: Feature history table

Feature Name	Release Information	Feature Description
BGP Prefix Origin Validation Based on RPKI	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
BGP Prefix Origin Validation Based on RPKI	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only); Modular Systems (8800 [LC ASIC: P100])(select variants only*) This feature enhances BGP route security by using Resource Public Key Infrastructure (RPKI) to validate the origin Autonomous System (AS). It associates a route's address prefix with AS numbers, starting with the origin AS, and uses RPKI to verify the AS claiming the prefix. This helps prevent prefix mis-announcement, ensuring routes are secure and legitimate. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Overview of RPKI

RPKI is a globally distributed cryptographic framework that creates a verifiable database of IP address blocks and Autonomous System (AS) numbers. RPKI enables network operators to securely certify which AS is authorized to advertise specific IP address prefixes.

BGP origin validation process

When a BGP router receives a route, it examines the AS_PATH attribute, which lists the sequence of ASes across which a prefix announcement has traveled. The router identifies the origin AS for the prefix and checks this against the authorization records in the RPKI database. If RPKI lists the prefix-origin pair as valid, the BGP router accepts the route. If not, the router considers the route invalid or suspicious. This process helps you prevent unauthorized or accidental route advertisements.

Security benefits

By leveraging RPKI-based BGP prefix origin validation, networks can defend against several well-known routing threats, such as prefix hijacking, mis-announcements, and monkey-in-the-middle attacks. Only routes authenticated through RPKI are trusted, reducing the risk of disrupted or maliciously re-routed Internet traffic.

Example of BGP prefix origin validation using RPKI

Suppose AS 64500 originates the prefix 192.0.2.0/24 and announces it into BGP. Another router receives this route and checks the RPKI database. If the database confirms that AS 64500 is authorized to originate 192.0.2.0/24, the route is considered valid. If not, the route is rejected or marked as suspicious, thus preventing a possible hijack or mis-announcement.

Configure an RPKI cache server

RPKI helps prevent route hijacking by verifying that BGP routes are correctly originated. Configuring a cache server allows the router to obtain validated prefix information for secure routing decisions.

Before you begin

- Obtain the RPKI cache server's IP address or hostname and transport requirements (SSH or TCP).
- Have SSH credentials available if using SSH as the transport protocol.

Procedure

Step 1 Enter RPKI cache server configuration mode and configure the transport protocol (TCP or SSH) and port for the cache server.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# rpki server 10.2.3.4
Router(config-bgp-rpki-server)# transport ssh port 22
```

Note
The default SSH port is 22. Both SSH and TCP support ports in the range 1–65535.

Tip
You can set the transport to either TCP or SSH. Changing the transport method causes the cache session to flap.

Step 2 (Optional, when using SSH) Set the username and password for the cache server.

Example:

```
Router(config-bgp-rpki-server)# username ssh_rpki_cache
Router(config-bgp-rpki-server)# password ssh_rpki_pass
```

Step 3 (Optional) Configure the preference for this cache server if multiple servers are used.

Example:

```
Router(config-bgp-rpki-server)# preference 1
```

Range for the preference value is 1 to 10. Lower values have higher priority.

Step 4 (Optional) Set the purge time for how long BGP retains route information after the cache session drops.

Example:

```
Router(config-bgp-rpki-server)# purge-time 30
```

Range for the purge time is 30 to 360 seconds.

Step 5 (Optional) Configure periodic refresh and response timers.

To set the refresh interval or disable it:

```
Router(config-bgp-rpki-server)# refresh-time 20
```

Or

```
Router(config-bgp-rpki-server)# refresh-time off
```

To set the maximum response wait time or disable the timeout:

```
Router(config-bgp-rpki-server)# response-time 30
```

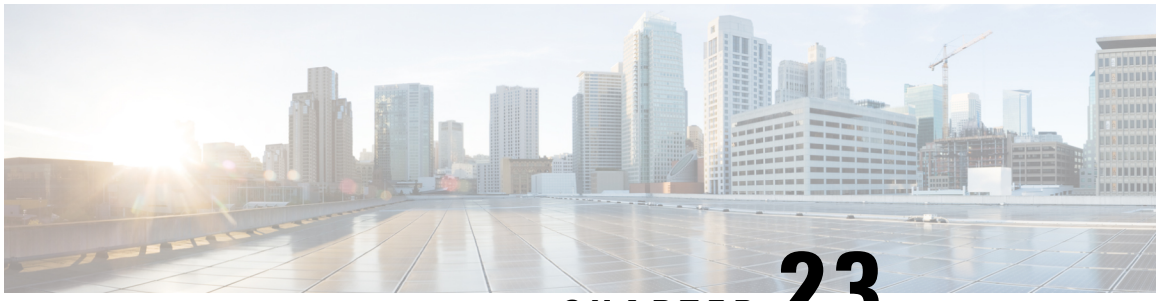
Or

```
Router(config-bgp-rpki-server)# response-time off
```

Step 6 (Optional) Shut down the RPKI cache server.

Example:

```
Router(config-bgp-rpki-server)# shutdown
```



CHAPTER 23

Monitoring and Debugging

This chapter offers a comprehensive guide to BGP labeled unicast, detailing its various implementations, including over RSVP-TE for traffic engineering and resiliency. It also covers the exclusion of label allocation for non-advertised routes and steering BGP control-plane traffic over IP-only paths to optimize MPLS network operations.

BGP-RIB feedback mechanisms for update generation

The BGP-RIB feedback mechanisms for update generation are BGP route control mechanisms that

- ensure routes are installed locally before advertising them to neighbors
- track which route versions the forwarding information base (FIB) has consumed, and
- send updates only for routes confirmed installed in the FIB to prevent premature advertisements.

These mechanisms help you avoid traffic loss caused by premature route advertisements after events like router reloads, line card online insertion and removal (LC OIR), or link flaps when alternate paths become available

You configure BGP to wait for routing information base (RIB) feedback before sending updates by using the **update wait-install** command in the `router address-family IPv4` or `router address-family VPNv4` configuration mode. This command ensures that BGP sends updates only after routes are confirmed installed in the forwarding information base (FIB), preventing premature route advertisements.

You can verify this configuration using the following commands:

- **show bgp**
- **show bgp neighbors**

- **show bgp process performance-statistics**

This configuration helps you avoid traffic loss caused by premature route advertisements after events such as router reloads, line card online insertion and removal (LC OIR), or link flaps when alternate paths become available.

Guidelines for BGP-RIB feedback mechanisms

To prevent traffic loss and ensure reliable routing, always advertise BGP routes only after they are confirmed as installed in the Forwarding Information Base (FIB) via the BGP-RIB feedback mechanism.

- ensure that BGP installs routes in the Routing Information Base (RIB) and waits for feedback from the RIB about installation in the FIB.
- confirm that the RIB tracks which route versions are in the FIB using the BCDL feedback mechanism.
- send BGP update messages only for routes confirmed as installed in the FIB, preventing premature advertisements that could cause packet loss or blackholing.

Configure BGP to wait for RIB feedback before sending updates

Enable BGP to delay advertising updates until routes are confirmed as installed in the FIB, preventing premature updates and possible traffic loss.

Use this configuration to enhance BGP routing reliability by ensuring updates are sent only after successful RIB-to-FIB installation confirmation.

Before you begin

- Verify you have administrator access to the router.
- Identify the AS number and desired address family (e.g., IPv4 unicast, VPNv4).

Follow these steps to configure BGP to wait for RIB feedback:

Procedure

Step 1 Enter router configuration mode for the desired address family.

Example:

```
Router# configure
Router(config)# router bgp 1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# update wait-install
```

Step 2 Save the configuration.

Example:

```
Router#(config-bgp-af)# commit
```

Step 3 Use the **show bgp process** command to view the delay of the BGP process update since the last router reload.

Example:

```
Router# show bgp process
Wed Aug 24 00:40:48.649 PDT

BGP Process Information:
BGP is operating in STANDALONE mode
Autonomous System number format: ASPLAIN
Autonomous System: 100
Router ID: 192.168.0.2 (manually configured)
Default Cluster ID: 192.168.0.2
Active Cluster IDs:  192.168.0.2
------------------------------
------------------------------
Update wait-install enabled:
  ack request 2, ack rcvd 2, slow ack 0
  startup delay 10 secs
```

BGP on the router now delays advertising route updates until RIB confirms that routes are installed in the FIB, ensuring reliable route propagation and preventing traffic loss.

What to do next

Monitor BGP updates and verify network stability after applying the configuration.

Enhanced monitoring of NSR statistics

The Enhanced monitoring of non-static routing (NSR) statistics is a network monitoring feature that

- provides detailed packet processing metrics during critical network operations,
- tracks BGP update processing speed, traffic volume, and packet sequence number integrity, and
- enables early detection of packet loss or sequencing inconsistencies to maintain routing reliability and network stability.

This feature helps you ensure continuous and reliable BGP routing by synchronizing state information between primary and standby routing engines during software upgrades or failover events. It gives you real-time insights into NSR activities so you can proactively address issues before they impact network performance.

NSR synchronization and monitoring for network stability and failover

- NSR synchronizes routing and forwarding data between primary and standby routing engines to minimize disruption during failover.
- Monitoring NSR statistics enables proactive identification and resolution of potential BGP routing issues.
- The feature supports comprehensive data analysis during pivotal network events, enhancing overall network stability and reliability.

Table 54: Feature History Table

Feature Name	Release Name	Description
Enhanced Monitoring of NSR Statistics	Release 24.2.1	You can maintain uninterrupted network functionality during upgrades or failovers with Non-Stop Routing (NSR), ensuring consistent data across primary and standby engines. The Enhanced Monitoring of NSR Statistics feature offers metrics on NSR packet handling, providing processing times, counts, and sequence numbers in real-time. If no new packets are received, the last known statistics persist, keeping the displayed data current. CLI: The feature modifies the output of the show command given below: • **show bgp nsr** YANG Data Model: • `Cisco-IOS-XR-ipv4-bgp-oper` (see GitHub, YANG Data Models Navigator)

View enhanced NSR statistics

Obtain and analyze key metrics to assess the efficiency and robustness of BGP operations with NSR functionality.

Use this task to monitor BGP update processing speed, packet volume over intervals, and sequence number integrity to prevent packet loss and maintain network stability.

Before you begin

Ensure you have access to the device CLI and necessary permissions to run show commands.

Follow these steps to view enhanced monitoring of NSR statistics:

Procedure

Verify the key metrics to evaluate BGP operations with NSR. Monitor BGP update processing speed, packet volume over intervals, and sequence number integrity to prevent packet loss and ensure network stability.

Example:

```
Router# show bgp nsr

Fri Jan 30 10:18:48.171 PST PDT

BGP Process Information:
BGP is operating in STANDALONE mode
Autonomous System: 100
Router ID: 10.1.0.1 (manually configured)
Default Cluster ID: 10.1.0.1
Active Cluster IDs:  10.1.0.1
Fast external fallover enabled
Neighbor logging is not enabled
Enforce first AS enabled
AS Path ignore is enabled
AS Path multipath-relax is enabled
Default local preference: 100
Default keepalive: 60
Graceful restart enabled
Restart time: 180
Stale path timeout time: 360
RIB purge timeout time: 600
Non-stop routing is enabled
Update delay: 120
Generic scan interval: 60

Address family: IPv4 Unicast
Dampening is not enabled
Client reflection is enabled in global config
Scan interval: 60
Main Table Version: 7034
IGP notification: IGPs notified
RIB has converged: version 1

========== Post Failover Summary for Active instance ==========

Node                Process            Read      Write    Inbound


node0_0_CPU0        Speaker          146.75      18.90       3.46

  Entered mode  Standby Ready               : Jan 30 10:00:39
  Entered mode  TCP NSR Setup               : Jan 30 10:00:39
  Entered mode  TCP NSR Setup Done          : Jan 30 10:00:39
  Entered mode  TCP Initial Sync            : Jan 30 10:00:39
  Entered mode  TCP Initial Sync Done       : Jan 30 10:00:44
  Entered mode  FPBSN processing done       : Jan 30 10:00:44
  Entered mode  Update processing done      : Jan 30 10:00:44
  Entered mode  BGP Initial Sync            : Jan 30 10:00:44
  Entered mode  BGP Initial Sync done       : Jan 30 10:00:44
  Entered mode  NSR Ready                   : Jan 30 10:00:44

Current BGP NSR state - NSR Ready achieved at: Jan 30 10:00:44
NSR State READY notified to Redcon at: Jan 30 10:16:58

NSR Post Failover Summary:

NPL Statistics:

Messages Sent: 384985 .           ACKS Received: 384985 :8
Messages Sent: 8                  ACKS Sent: 8
Send failures: 11541 .            Send ACK Failures:0
Suspends:      11541              Resumes: 11407
Messages Processed: 8             Out of sequence drops: 8
Messages Send Drops: 0            Messages Recv Drops: 0
```

```
Sync Send Timeouts: 8

NPL Packet Processing Statistics:

  Interval       End-Time        Avg Proc  Num of        seq num
   (sec)                         time(us)   pkts     [start - end]

    30     Aug 22 23:08:11.142     140         2     [     74 - 75     ]
    60     Aug 22 23:08:11.142     233         4     [     72 - 75     ]
   180     Aug 22 23:08:11.142     426        22     [     54 - 75     ]


QAD Statistics:

  Messages Sent       : 512           ACKs Received       : 512
  Messages Received   : 8             ACKs Sent           : 8
  Send Failures       : 1             Send ACK Failures   : 0
  Suspends            : 1             Resumes             : 1
  Messages Processed  : 8             Out of sequence drops: 0
Postit Summary:

  Total pending postit messages:  0
  Neighbors with pending postits: 0

Conv  Bestpath   TunnelUpd  Import     RIBUpd     Label      ReadWrite  LastUpd
Process: Speaker

Yes   120        ---        ---        120        120        120        87531

Rib Trigger: enabled
Last RIB down event  Jan 29 09:50:03.069 received
Last RIB convergence Jan 29 09:50:03.069 last ack received.

Address Family IPv4 Unicast converged in 87531 seconds
```

Review the output to conform the detailed metrics on BGP NSR packet handling over specific time intervals.

You obtain detailed NSR health data to prevent packet loss and maintain reliable network operations during critical events.

Store and analyze changes in the prefixes received from BGP peer

The store and analyze changes in the prefixes received from BGP peer is a serviceability feature that

- allow routers to handle analysis of millions of BGP paths across IPv4 and IPv6 address families,
- enables operators to actively monitor and analyze changes, acceptances, and rejections of prefixes received from BGP peers by providing detailed statistics, and
- allows you to store all original copies of routes received from peers, including those not selected as the best path.

This feature helps you improve serviceability by monitoring BGP operations and facilitating debugging in both production and lab environments.

How you use this feature

- Soft reconfiguration stores incoming prefixes before applying policies if the peer does not support route refresh; using the **always** keyword forces storage even when route refresh is supported.
- The **soft reconfiguration inbound always** command enables storage of all updates from a specified neighbor.
- The **soft reconfiguration inbound** command stores the original unmodified route alongside modified or filtered routes, allowing you to perform a "soft clear" after inbound policy changes.
- Prefixes fall into three categories: accepted and unmodified, accepted and modified, or denied.
- Operators can monitor the impact of inbound policies without network disruption, helping to maintain stability and facilitate troubleshooting.

Verify the BGP prefix statistics

Ensure that you verify the soft reconfiguration statistics and evaluate the impact of inbound policies on IPv4 unicast BGP sessions to maintain proper routing behavior and policy compliance.

Before you begin

- Confirm that BGP is configured and that soft reconfiguration is enabled on the relevant neighbors.
- Verify that you know the IP addresses of BGP neighbors and the inbound policies applied.
- Be aware that soft reconfiguration is memory intensive; ensure the router has sufficient resources.
- Understand that the `soft-reconfiguration inbound` command stores updates from neighbors, enabling soft resets even if the neighbor does not support route refresh.

Procedure

Step 1 Use the **show bgp ipv4 unicast summary soft-reconfig-stats**command to verify the soft reconfiguration statistics for IPv4 unicast BGP sessions.

Example:

```
Router# show bgp ipv4 unicast summary soft-reconfig-stats

....
.....

Neighbor     Spk  AS  MsgRcvd MsgSent TblVer InQ  OutQ  Up/Down     St/PfxRcd  SoftChgd  Denied
10.10.10.4    0   3    15       12      0    0     0    00:46:06 2           0       0
Total                                                             2           0       0

Legend:

Total PfxRcd: Sum of accepted unmodified and modifed paths
Total SoftChgd: Sum of accepted modified paths
Total Denied: Sum of Denied paths
```

This output displays the soft reconfiguration statistics for IPv4 unicast BGP sessions given below:

- Total PfxRcd represents the sum of accepted unmodified and modified paths.

- Total SoftChgd represents the sum of accepted modified paths.
- Total Denied represents the sum of denied paths.

Step 2 Use the **show bgp ipv4 unicast neighbors dryrun-policy pass** command to verify the dry run policy impact on inbound policy.

Example:

```
Router# show bgp ipv4 unicast neighbors 10.10.10.1 dryrun-policy pass

Sat Oct 14 01:22:02.946 EDT

Policy Statistics
-----------------
     AFI:                         IPv4 Unicast
     Direction:                   Inbound
     In-use Policy:               pass
     Dry-run Policy:              pass
     Remote-as:                   300
     Total Networks walked:       257
     Total Paths walked:          72257
     Dry Run elapsed time(ms):    8
-------------------------------------------------------------------------------------
                                      Dry-run-Policy      In-use-Policy      Delta
-------------------------------------------------------------------------------------
Neighbor: 10.10.10.1
     Accepted Unmodified:             257                 257                0
     Accepted Modified:               0                   0                  0
     Pre-inbound policy copy:         0                   0                  0
     Denied:                          0                   0                  0
     Estimated Total Paths Memory:    26.10KB             26.10KB            0.00
-------------------------------------------------------------------------------------
    /* The values in the table provides information confirms that the BGP session with the
       neighbor 10.10.10.1 is passing the dry run policy.
       The values indicates that the BGP updates from the neighbor comply with the specified policies

      without actually applying the policies. The values in the table provides insight into potential

       routing changes without committing the policies. */
```

The table values confirm that the BGP session with neighbor 10.10.10.1 is successfully passing the dry run policy. This indicates that BGP updates from the neighbor comply with the specified policies without actually applying them. These values provide insight into potential routing changes without committing the policies, enabling you to evaluate policy impact safely.

Step 3 Use the **show bgp scale detail** command to verify the statistics on configured and established neighbors, and address-family prefixes, paths, and memory usage.

Example:

```
Router# show bgp scale detail
Fri Feb 2 12:49:38.349 EST

VRF: default
Neighbors Configured: 2 Established: 2

Address-Family  Prefixes  Paths  PathElem  Prefix       Path      PathElem
                                           Memory       Memory    Memory

IPv4 Unicast    3         5      3         564.00       520.00    369.00

SoftReconfig Changed  1  104.00 ---> This field shows that soft reconfiguration has been enabled. It
```

```
 also displays the number of prefixes that were accepted and modified, and the amount of memory
consumed by the prefix.
------------------------------------------------------------------------------
Total           3          5        3      564.00       520.00    369.00

Total VRFs Configured: 0
```

The field SoftReconfig Changed 1 104.00 indicates that soft reconfiguration is enabled. It also shows the number of prefixes that were accepted and modified, along with the memory consumed by these prefixes, providing insight into resource usage related to soft reconfiguration.

Monitor BGP memory statistics

The BGP memory statistics is a serviceability feature that:

- periodically monitors memory usage by the BGP process every 60 seconds,
- logs any memory changes exceeding 1% of the configured resource limit (rlimit), and
- triggers syslog notifications at 85%, 90%, and 95% of the rlimit to alert administrators before critical memory exhaustion.
- The **show bgp memory history** command is a diagnostic tool that displays memory usage trends and variations over time, allowing administrators to analyze historical memory consumption patterns.
- Syslog notifications are automated alerts generated when memory usage reaches predefined thresholds, enabling proactive management of memory resources before critical limits are reached.
- A serviceability feature is a functionality designed to maintain system health by providing monitoring, alerting, and diagnostic capabilities that support troubleshooting and capacity planning.

The **show bgp memory history** command displays detailed historical memory usage information, including total memory in megabytes, percent of rlimit used, memory differences between records, and counts of networks, paths, path elements, and attributes for the default VRF.

Monitor BGP Memory Statistics

This task instructs you to check the historical memory usage of the BGP process to monitor memory trends and manage resources proactively.

Procedure

Step 1 Use the **show bgp memory history** command on the active router to verify the BGP memory history:

Example:

```
Router# show bgp memory history

History of memory changes recorded for a threshold greater than 1.0% of rlimit.
Last shown record displays current values.
Network information for default VRF.
```

```
Time              Memory(MB)  Rlimit(%)  Memory diff(MB)  Networks  Paths     PathElems  Attributes

Oct  2 16:30:37    152         1          152              400       400       400        9
Oct  2 16:31:37    343         4          191              396952    396869    396952     725
Oct  2 16:32:37    425         5          81               524567    513979    524567     8408
Oct  2 16:42:38    741         9          316              1178605   1241533   1178604    10753
Oct  2 16:43:38    985         12         243              1778234   1859254   1778234    11214
Oct  2 19:42:39    901         11         -84              1800688   678607    1800688    10911
Oct  2 19:45:39    766         9          -136             1332259   688784    1332259    10943
```

The show command output displays memory change history for thresholds above 1.0% of the resource limit, with the latest record reflecting current values on the active router. It includes network details for the default VRF such as timestamps, memory in MB, rlimit percentage, memory differences, and counts of networks, paths, path elements, and attributes.

Step 2 Use the **show bgp memory history standby** command on the standby router to verify the BGP memory history:

Example:

```
Router# show bgp memory history standby
Sat Mar  2 00:26:46.874 UTC

History of memory changes recorded for a threshold greater than 1.0% of rlimit.
Last shown record displays current values.
Network information for default VRF.

Time              Memory(MB)  Rlimit(%)  Memory diff(MB)  Networks  Paths     PathElems  Attributes

Feb  9 03:39:04    98          1          98               0         0         0          0

Feb  9 03:42:04    2913        35         2814             2372674   14546789  2372674    170613

Feb  9 03:43:04    3129        38         216              2466877   16016399  2466877    181072

Feb  9 03:44:04    3310        40         180              2510788   17302274  2510788    190648

Feb  9 03:45:04    3601        43         291              2579305   19470841  2579305    210759

Feb  9 03:46:04    3825        46         224              2657361   20952659  2657361    240920

Feb  9 03:47:04    4063        49         238              2747506   22538284  2747506    262756

Feb  9 03:48:04    4298        52         234              2830363   24126386  2830363    284014

Feb  9 03:49:04    4530        55         231              2909578   25734085  2909578    304881

Feb  9 03:50:04    4753        58         222              2984782   27302279  2984782    324646

Feb  9 03:51:04    4961        60         208              3057329   28792696  3057329    342571

Feb  9 03:52:05    5177        63         215              3135909   30322183  3135909    360386

Feb  9 03:53:05    5393        65         216              3223111   31851898  3223111    377234

Feb  9 03:54:05    5550        67         156              3229253   33250926  3229253    382132

Feb  9 03:55:05    5694        69         143              3229253   34599173  3229253    385339

Feb  9 03:56:05    5832        71         138              3229253   35912290  3229253    387534

Feb  9 03:57:05    5987        73         155              3229257   37416025  3229257    389403

Feb  9 03:58:05    6133        74         145              3229257   38817868  3229257    390404
```

```
Mar  2 00:26:46    6248        76         114              3229257   39991732  3229257    390551
```

The show command output displays memory change history for thresholds above 1.0% of the resource limit, with the latest record reflecting current values on the standby router. It includes network details for the default VRF such as timestamps, memory in MB, rlimit percentage, memory differences, and counts of networks, paths, path elements, and attributes.

Enhanced monitoring of BGP keepalive messages

An enhanced monitoring of BGP keepalive messages is a network monitoring feature that

- manages per-neighbor input and output queues,
- triggers throttling when packet volume exceeds thresholds or TCP buffers are full, and
- monitors keepalive intervals and hold timers to track session stability.

This feature logs precise times when neighbors' queues enter and exit throttled states, records the maximum duration of throttling events, and maintains historical data including a circular buffer of the last 10 throttling states per neighbor that persists across resets. It also tracks parameters such as maximum hold time elapsed since the last keepalive message, timestamps of these events, and counts of hold time threshold crossings, which indicate session stability or potential issues.

- This monitoring helps you maintain network stability and fairness by controlling message processing rates.
- You can view detailed session stability and throttling statistics by running the **show bgp neighbor detail** command.

View enhanced monitoring of BGP keepalive messages

To view detailed information about BGP session stability and message handling to monitor keepalive message timing and throttling behavior.

The enhanced monitoring of BGP keepalive messages feature enables adminstrators to ensure network stability and fairness by managing BGP message processing rates. It provides key insights into session stability, throttling, and timer metrics essential for diagnosing and troubleshooting delayed or lost keepalive messages that affect BGP session continuity.

Before you begin

Before you begin, gather the necessary details to use the Enhanced Monitoring of BGP Keepalive Messages feature:

- Access to the router CLI with permissions to run commands, such as **show bgp neighbor detail**
- Configured and active BGP neighbors to monitor
- Knowledge of keepalive interval and hold timer settings for interpreting monitoring data

- Readiness to analyze throttling statistics, hold time events, and queue metrics for session stability and troubleshooting

Procedure

Use the **show bgp neighbor detail** command to view the data on BGP neighbors, including critical timers and queue metrics.

Example:

```
....
Max Hold Time elapsed was 6001 msec at Sep 12 17:02:36.954, crossed 40%: 2, 70%: 0

  Max Hold Time elapsed before reset was 9001 msec at Sep 12 17:01:53.397, crossed 40%: 7, 70%: 2

  First message received at Sep 12 16:45:00.973, sent at Sep 12 16:45:00.975

  First message before reset received at Sep 12 16:42:16.573, sent at Sep 12 16:42:16.574

  Max read throttled duration was 6769 msec starting at Sep 12 16:45:01.487, max InQ 1000 processed
 930

    Most recent read throttle periods (in msec):

    Start Time             Duration     Max InQ    Messages

    Sep 12 17:00:16.937          14         104          45

    Sep 12 17:00:16.954           9         136          74

    Sep 12 17:00:47.358          11         125         135

    Sep 12 17:01:02.658           2          83           0

    Sep 12 17:01:02.693           7         110           0

    Sep 12 17:01:02.705          13         139          74

    Sep 12 17:01:17.856           5          92          60

    Sep 12 17:01:17.877           3          91          30

    Sep 12 17:01:17.891          10         135          74

    Sep 12 17:01:33.128          21         132         193

  Max read throttled duration before reset was 5013 msec starting at Sep 12 16:42:17.079, max InQ 76
 processed 0

    Most recent read throttle periods before reset (in msec):

    Start Time             Duration     Max InQ    Messages

    Sep 12 16:42:17.079        5013          76           0

  Max write throttled duration was 685 msec starting at Sep 12 16:45:08.486, max OutQ 1501 queued 57


    Most recent write throttle periods (in msec):
```

```
    Start Time             Duration    Max OutQ    Messages

    Sep 12 17:01:38.799          46         398          23

    Sep 12 17:01:38.846         202         342          57

    Sep 12 17:01:39.049          47         320          23

    Sep 12 17:01:39.097         202         264          57

    Sep 12 17:01:39.299          46         242          22

    Sep 12 17:01:39.346         204         185          58

    Sep 12 17:01:39.551          45         164          23

    Sep 12 17:01:39.597         202         108          57

    Sep 12 17:01:39.799          46          86          23

    Sep 12 17:01:39.847         202          30           8

  Max write throttled duration before reset was 205 msec starting at Sep 12 16:42:21.849, max OutQ
1003 queued 1

    Most recent write throttle periods before reset (in msec):

    Start Time             Duration    Max OutQ    Messages

    Sep 12 16:42:21.849         205        1003           1

    Sep 12 16:42:22.055          20         925          23

    Sep 12 16:42:22.075          16         869          56
....
```

This table shows key fields from the show bgp neighbor detail output related to enhanced BGP keepalive monitoring, including timers, throttling, queue sizes, and message stats for assessing session stability.

This table describes the significant fields shown in the display.

Table 55: Fields pertaining to enhanced monitoring of BGP keepalive messages in the output of show bgp neighbor details command

Field	Description
Max Hold Time elapsed	Maximum amount of time that has passed since the last BGP keepalive message was received from a neighbor before a BGP session is considered to be down.

Field	Description
Max Hold Time elapsed was 6001 msec at Sep 12 17:02:36.954, crossed 40%: 2, 70%: 0	Maximum amount of time that has passed since the last BGP keepalive message was received from a neighbor before a BGP session is considered to be down. In this specific output, the fields indicate the following: Max Hold Time elapsed was 6001 msec: indicates that the maximum time interval between receiving keepalive messages from the neighbor was 6001 milliseconds or approximately 6 seconds. at Sep 12 17:02:36.954: Timestamp when this maximum hold time was observed. crossed 40%: 2, 70%: 0: Number of times the hold time crossed certain thresholds. The hold time crossed the 40% threshold twice and the 70% threshold zero times, suggesting that the hold time reached a significant portion of its configured value but did not exceed it by a large margin.
Max Hold Time elapsed before reset was 9001 msec at Sep 12 17:01:53.397, crossed 40%: 7, 70%: 2	Maximum duration between receiving BGP (Border Gateway Protocol) keepalive messages from a neighbor before the BGP session was reset. In this specific output, the fields indicate the following: Max Hold Time elapsed before reset was 9001 msec: Maximum time interval between receiving keepalive messages from the neighbor before the BGP session reset was 9001 milliseconds or approximately 9 seconds. at Sep 12 17:01:53.397: Timestamp when this maximum hold time before reset was observed. crossed 40%: 7, 70%: 2: Number of times the hold time crossed certain thresholds. The hold time crossed the 40% threshold seven times and the 70% threshold two times, suggesting that the hold time frequently approached significant portions of its configured maximum value before the BGP session reset
First message received at Sep 12 16:45:00.973, sent at Sep 12 16:45:00.975	Timestamp when the first message from a BGP neighbor was received by the local router. In this specific output, the fields indicate the following: First message received at Sep 12 16:45:00.973: First message from the BGP neighbor was received at 16:45:00 on September 12th sent at Sep 12 16:45:00.975: Timestamp when the corresponding message was sent by the BGP neighbor, which was nearly simultaneously, just 0.002 seconds later.
First message before reset received at Sep 12 16:42:16.573, sent at Sep 12 16:42:16.574	Timestamp when the first message from a BGP neighbor was received by the local router before a reset occurred. In this specific output, the fields indicate the following: First message before reset received at Sep 12 16:42:16.573: first message from the BGP neighbor was received at 16:42:16 on September 12th, before a reset occurred. sent at Sep 12 16:42:16.574: Timestamp when the corresponding message was sent by the BGP neighbor, which was nearly simultaneous, just 0.001 seconds later.

Field	Description
`Max read throttled duration was 6769 msec starting at Sep 12 16:45:01.487, max InQ 1000 processed 930`	Maximum duration during which the read process was throttled, indicating a restriction or limitation on the rate of reading data. In this specific output, the fields indicate the following: Max read throttled duration was 6769 msec: Maximum duration of throttling for reading data was 6769 milliseconds (approximately 6.769 seconds). starting at Sep 12 16:45:01.487: Timestamp when this maximum throttling duration started, which was at 16:45:01 on September 12th. max InQ 1000 processed 930: Maximum input queue (InQ) size was 1000, and during the throttled duration, 930 items were processed.
Start Time	Timestamp when the read throttle period started.
Dry Run elapsed time(ms)	Time taken for the dry run in milliseconds.
Duration	Duration of the throttle period in milliseconds, indicating how long the read process was restricted or limited.
Max InQ	Maximum size of the input queue during the throttle period. The input queue typically holds incoming data packets waiting to be processed.
Messages	Number of messages or data packets processed during the throttle period.
Max read throttled duration before reset was 5013 msec starting at Sep 12 16:42:17.079, max InQ 76 processed 0	Maximum duration of a read throttle period on the network device. In this specific output, the fields indicate the following: Max read throttled duration before reset: Maximum duration of the read throttle period, which was 5013 milliseconds or approximately 5.013 seconds. Starting at Sep 12 16:42:17.079: Timestamp when the read throttle period started, which was at 16:42:17 on September 12th *Max InQ 76 processed 0*: The segment *Max InQ 76* indicates that the maximum size of the input queue during the throttle period was 76. The segment *processed 0* indicates that no messages or data packets were processed during this throttle period.
Max write throttled duration was 685 msec starting at Sep 12 16:45:08.486, max OutQ 1501 queued 57	Maximum duration of the write throttle period, which was 685 milliseconds. In this specific output, the fields indicate the following: Max write throttled duration: Maximum duration of the write throttle period, which was 685 milliseconds. Starting at Sep 12 16:45:08.486: Timestamp when the write throttle period started, which was September 12th at 16:45:08.486. Max OutQ: Maximum size of the output queue during the throttle period. In this case, it was 1501, which typically holds data packets waiting to be transmitted. Queued: Number of items queued in the output queue during the throttle period. In this case, it was 57.

Field	Description
Max write throttled duration before reset was 205 msec starting at Sep 12 16:42:21.849, max OutQ 1003 queued 1	Maximum duration of a write throttle period on a network device before a reset occurred. In this specific output, the fields indicate the following: Max write throttled duration before reset: Maximum duration of the write throttle period before a reset occurred, which was 205 milliseconds. Starting at Sep 12 16:42:21.849: Timestamp when the write throttle period started, which was on September 12th at 16:42:21.849. Max OutQ: Maximum size of the output queue during the throttle period. In this case, it was 1003, indicating the maximum number of items that were waiting to be transmitted. Queued: Number of items queued in the output queue during the throttle period. In this case, it was 1.
Start Time:	Timestamp when the write throttle period started.
Duration	Duration of the write throttle period in milliseconds.
Max OutQ	Maximum size of the output queue during the throttle period. The output queue typically holds data packets waiting to be transmitted.
Messages	Number of messages or data packets transmitted during the throttle period.

Enhanced monitoring of BGP memory utilization

Enhanced monitoring of BGP memory utilization is a network monitoring feature that

- tracks memory usage by the BGP process within a device,
- provides periodic memory state checks and logs changes, and
- triggers notifications when memory usage approaches critical thresholds.

This feature serves as an early warning system to help you maintain network stability and performance by proactively managing BGP memory consumption.

Key features of enhanced monitoring of BGP memory utilizations:

- Periodic Memory State Check: The system automatically monitors memory usage every 60 seconds to detect issues promptly while minimizing overhead.
- Logging Mechanism: Memory changes are recorded in both the BGP trace log and a circular buffer, ensuring reliable data for troubleshooting.
- Significant Change Detection: Any increase in memory usage exceeding 1% of the configured resource limit (rlimit) since the last report is logged to identify trends or spikes.

- Resource Limit (rlimit): This predefined threshold limits the maximum memory BGP can use to prevent system instability caused by excessive memory consumption.
- Syslog Notifications: Notifications are sent to syslog at 85%, 90%, and 95% of the resource limit, escalating in frequency as memory usage nears the limit to alert administrators effectively.

Monitor BGP memory utilization

Track BGP memory usage changes when utilization exceeds 1% of the configured resource limit (rlimit) to maintain network stability.

Use the **show bgp memory history** command to review BGP memory usage changes exceeding 1% of the resource limit for proactive monitoring.

Before you begin

Make sure you have appropriate access rights to run the **show bgp memory history** command on the device.

Follow these steps to monitor BGP memory utilization:

Procedure

Step 1 Use the **show bgp memory history** command to view the data on memory utilization on the active router.

Example:

```
Router# show bgp memory history
History of memory changes recorded for a threshold greater than 1.0% of rlimit.
Last shown record displays current values.
Network information for default VRF.

Time               Memory(MB)  Rlimit(%)  Memory diff(MB)  Networks  Paths     PathElems  Attributes

Oct  2 16:30:37    152         1          152              400       400       400        9
Oct  2 16:31:37    343         4          191              396952    396869    396952     725
Oct  2 16:32:37    425         5          81               524567    513979    524567     8408
Oct  2 16:42:38    741         9          316              1178605   1241533   1178604    10753
Oct  2 16:43:38    985         12         243              1778234   1859254   1778234    11214
Oct  2 19:42:39    901         11         -84              1800688   678607    1800688    10911
Oct  2 19:45:39    766         9          -136             1332259   688784    1332259    10943
```

This command displays memory changes exceeding 1.0% of rlimit, with the last record showing current values and network information for the default VRF on the active router.

Step 2 Use the **show bgp memory history standby** command to view the data on memory utilization on a standby router.

Example:

```
Router# show bgp memory history standby
History of memory changes recorded for a threshold greater than 1.0% of rlimit.
Last shown record displays current values.
Network information for default VRF.

Time               Memory(MB)  Rlimit(%)  Memory diff(MB)  Networks  Paths     PathElems  Attributes

Feb  9 03:39:04    98          1          98               0         0         0          0
```

```
Feb  9 03:42:04    2913        35         2814            2372674   14546789  2372674    170613
Feb  9 03:56:05    5832        71         138             3229253   35912290  3229253    387534
Feb  9 03:57:05    5987        73         155             3229257   37416025  3229257    389403
Feb  9 03:58:05    6133        74         145             3229257   38817868  3229257    390404
Mar  2 00:26:46    6248        76         114             3229257   39991732  3229257    390551
```

This command displays memory changes exceeding 1.0% of rlimit, with the last record showing current values and network information for the default VRF on the standby router.

This table shows key fields from the show bgp memory history details output related to enhanced BGP memory utilization.

Table 56: Fields pertaining to enhanced monitoring of BGP memory utilization in the output of show bgp memory history details command

Field	Description
Time	Timestamp when the rmeasurement was taken.
Memory (MB)	Total memory in megabytes (MB) used by the routing process at the specified time.
Rlimit (%)	Percentage of the memory resource limit that is being used.
Memory diff (MB)	Quantity of memory usage in megabytes (MB) that has increased or decreased since the last report.
Networks	Number of network prefixes known to the router.
Paths	Number of distinct paths to various destinations.
PathElems	Number of path elements (such as AS numbers) involved in routing.
Attributes	Number of unique BGP attributes in use, such as local preference, and MED.

You obtain a detailed historical view of BGP memory usage for proactive monitoring of resource consumption.

Verify enhanced next hop monitoring

Use these commands to monitor BGP next-hop reachability, metric changes, and event counters to analyze routing dynamics and network stability.

This feature helps you promptly identify and analyze routing path changes, which is essential for maintaining network stability and performance, especially in large-scale environments. By using specific show commands, you can obtain detailed insights into next-hop status, event counters, and historical event data, enabling effective troubleshooting and optimization of routing decisions.

Before you begin

Before you begin, make sure that you have proper access rights and privileges to run BGP-related show commands on the router.

Follow these steps to monitor BGP enhanced next hop monitoring:

Procedure

Step 1 Use the **show bgp nexthops** command to verify the details of nexthop reachability and metric change counters.

Example:

```
Router# show bgp nexthops
....
Next Hop        Reachable   Unreachable   MetricIncrease   MetricDecrease

0.0.0.0
10.10.10.1              1             0                0                0
203.0.113.1             2             1                0                0
192.168.0.3             1             0                1                2
192.168.0.5             1             0                0                0

.....
```

Step 2 Use the **show bgp nexthops wide** command to verify detailed information about BGP next-hop processing times, status codes, event counters, and metrics for each gateway address family.

Example:

```
Router# show bgp nexthops wide
....
Next Hop             Status      Metric    Tbl-ID        Notf      LastRIBEvent    RefCount     R
  U    MI    MD

0.0.0.0                                                                              25/3
10.10.10.1             [R][C][NL]       0      e0000000         1/0    00:14:52 (Cri)    17/20
 1     0     0     0
203.0.113.1            [R][NC][NL]      2      e0000000         0/0    00:02:06 (Reg)     5/7
 1     0     0     0
192.168.0.3            [R][NC][NL]      3      e0000000         0/0    00:02:06 (Reg)    12/246
 1     0     0     0
192.168.0.5            [R][NC][NL]      2      e0000000         1/0    00:14:17 (Cri)    16/270
 1     0     0     0

.....
```

Step 3 Use the **show bgp nexthops** *ipaddress* command to verify detailed BGP next-hop information for the IP address 10.10.10.1, including VRF, nexthop ID, flags, advertising neighbors, RIB details, event history, and reference counts.

Example:

```
Router# show bgp nexthops wide
.....
Reachable Notifications:        2 (last at Sep 11 16:04:56.738)
     Unreachable Notifications:         1 (last at Sep 11 16:04:36.520)
     Metric Increase Notifications:     2
     Metric Decrease Notifications:     1
     Most Recent Events:
      Time                      Event Type      Metric
      Sep 11 16:04:36.520       Unreachable          -
      Sep 11 16:04:56.738       Reachable            2
      Sep 11 16:30:38.402       Reachable           21
      Sep 11 16:31:23.548       Reachable           16
      Sep 11 16:34:59.460       Reachable          101
.....
```

This table shows key fields from the **show bgp nethops** output related to enhanced next hop monitoring, including next-hop IP address, status, metric, notification counts, last RIB event time, reference counts, and event counters for reachability and metric changes, enabling effective tracking and optimization of routing path stability in large-scale networks

Table 57: Fields pertaining to Enhanced Next Hop Monitoring in the output of show bgp nexthops command

Next Hop	The IP address of the next-hop router in the BGP network.
Status	A set of codes indicating the reachability and other status details about the next hop (e.g., Reachable, Unreachable, etc.).
Metric	The metric value used by BGP to determine the best path to the next hop. Lower values are preferred.
Tbl-ID	The unique identifier for the table in which the next-hop information is stored.
Notf	Notifications received/sent related to the next hop, often indicating BGP updates or state changes
LastRIBEvent	The time elapsed since the last Routing Information Base (RIB) event that pertained to this next hop.
RefCount	Reference count, which can indicate how many routes are using this next hop.
R (Reachable)	Event counter for the number of times the next hop has been marked as reachable.
U (Unreachable)	Event counter for the number of times the next hop has been marked as unreachable.
MI (Metric Increased)	Metric value for a particular route that has increased compared to the previous metric value.
MI (Metric Decreased)	Metric value for a particular route that has decreased compared to the previous metric value.
Reachable Notifications	The number of times a route has become reachable, that is a valid route to a destination is available, and the time of the last such notification.
Unreachable Notifications	The number of times a route has become unreachable, that is a previously valid route is no longer available, and the time of the last such notification.
Metric Increase Notifications	The number of times the metric for a route has increased, which typically makes the route less preferred.
Metric Decrease Notifications	The number of times the metric for a route has decreased, which usually makes the route more preferred.

Most Recent Events	List of individual routing events, including the time they occurred, the type of event, and the metric associated with the event. This also indicates the relative desirability of the route, with lower metrics being more preferred.
Unreachable	Indicates a loss of route.
Reachable	Indicates that a route is available.

You have verified BGP next-hop reachability and analyzed event counters, supporting efficient routing stability assessment and troubleshooting.

Enhanced next hop monitoring

Enhanced next hop monitoring is network monitoring feature that

- track changes in BGP next hop metrics and reachability,
- provide detailed event data including counters and timestamps, and
- aggregate statistics globally to assess network health dynamically.

This feature closely integrates BGP with the Routing Information Base (RIB) to receive notifications about next hop status changes, which trigger recalculations of optimal routing paths. This monitoring helps you identify root causes of network variations, especially during high routing activity or instability, which can strain CPU resources.

These key monitoring functions provide detailed insights into next hop status changes and overall network health, enabling effective tracking and analysis:

- Event counter tracking: Counts key events per next hop such as reachable, unreachable, metric increases, and decreases.
- Event history logging: Records the last five events with timestamps for each next hop.
- Recent event tracking: Logs transitions of next hops to reachable or unreachable states with precise timing.
- Global aggregation: Summarizes event counters by Address Family Identifier (AFI) for overall network health insight.
- Temporal counter analysis: Continuously updates event counts and reports changes over 1, 3, and 5-minute intervals, providing a dynamic view of network stability and routing effectiveness.

By using these monitoring capabilities, you can proactively manage BGP next hop changes, improving network reliability and performance.

Enhanced monitoring of version-rate statistics

The enhanced monitoring of version-rate statistics is a BGP monitoring feature that

- identifies sources of BGP churn by tracking and categorizing version bumps,
- calculates version rates over fixed time intervals, and
- generates detailed reports at both Address-Family Identifier (AFI) and neighbor AFI levels.

This feature also maintains a cumulative churn count and classifies version bumps by their origin, such as reachable, unreachable, import, redistribution, or label-related sources.

Additional details of enhanced monitoring of version-rate statistics

The following details provide additional information about how version-rate statistic monitoring features operate:

- Interval mechanism: Version bumps are monitored using fixed time intervals rather than sliding windows. For example, if an interval runs from 12:00 pm to 12:30 pm, data updates occur exactly at 12:30 pm; the next interval starts immediately after.
- NSR synchronization: Both active and standby routers independently record version bumps. Therefore, statistics may differ between active and standby routers.
- Reporting parameters: Reports are generated at both AFI and neighbor AFI levels. These reports include totals and interval-specific rates, categorized into buckets such as reachable, unreachable, import, redistribute, and label. Additional version bumps from other sources are also displayed, based on the main table's version number.

Enhanced monitoring of version-rate statistics

Enable monitoring and analysis of BGP version changes over specified fixed intervals for diagnostic and troubleshooting purposes.

This task helps you analyze BGP version changes over fixed time intervals, categorizing them by neighbor IP, VRF, and address family (AFI). Use these steps to gain visibility into the frequency and source of version changes, which may indicate network events, configuration changes, or route instabilities.

Before you begin

Ensure you have appropriate access rights to run the **show bgp memory history** command on the device.

Follow these steps to monitor BGP version-rate statistics:

Procedure

Step 1 Use the **show bgp sessions version-rate** command to o obtain a detailed analysis of BGP version changes across specified intervals;

Example:

```
Router# show bgp sessions version-rate
Thu Nov  2 11:40:41.107 IST
Interval definition(s):
  Interval   Duration (min)  Start time            End time
  Live                   5  Nov  2 11:37:22.029   Nov  2 11:40:41.413
  1                      5  Nov  2 11:32:22.029   Nov  2 11:37:22.029
  2                     15  Nov  2 11:22:22.029   Nov  2 11:37:22.029
```

```
  3                    30  Nov  2 10:52:22.027   Nov  2 11:22:22.027

Neighbor    VRF        AFI                 Total          Live
                                                        Total  Reach  UnReach
-------------------------------------------------------------------------------
10.10.10.1     default    All                   5    0      0        0
10.10.10.1                IPv4 Unicast          5    0      0        0
192.168.0.5    default    All                 606    0      0        0
192.168.0.5               IPv4 Unicast         63    0      0        0
192.168.0.5               VPNv4 Unicast       240    0      0        0
192.168.0.5               IPv6 Labeled-unicast 63    0      0        0
192.168.0.5               VPNv6 Unicast       240    0      0        0
192.168.0.5               RT Constraint         0    0      0        0
10:10:10::1    default    All                   5    0      0        0
10:10:10::1               IPv6 Unicast          5    0      0        0
10.0.1.1       1          All                   5    0      0        0
10.0.1.1                  IPv4 Unicast          5    0      0        0
10:0:1::1      1          All                   5    0      0        0


/* The output was too wide, so it was segmented; the below segment continues from above.
 */
 Interval 1              Interval 2            Interval 3  Spk AS
Total  Reach UnReach Total Reach UnReach Total Reach  UnReach
-------------------------------------------------------------------
  5      5       0     5     5      0       0     0      0      0   200
  5      5       0     5     5      0       0     0      0
282    282       0   282   282      0       0     0      0      0   100
 31     31       0    31    31      0       0     0      0
110    110       0   110   110      0       0     0      0
 31     31       0    31    31      0       0     0      0
110    110       0   110   110      0       0     0      0
  0      0       0     0     0      0       0     0      0
  5      5       0     5     5      0       0     0      0      0    200
  5      5       0     5     5      0       0     0      0
  0      0       0     0     0      0       0     0      0      0    200
  0      0       0     0     0      0       0     0      0
  0      0       0     0     0      0       0     0      0      0    200

/* The output was too wide, so it was segmented; the below segment continues from above.
 */

InQ OutQ NBRState     NSRState
----------------------------
0   0    Established  NSRReady

0   0    Established  NSRReady

0   0    Established  NSRReady

0   0    Established  NSRReady

0   0    Established  NSRReady
```

The output displays interval definitions, neighbor information (IP, VRF, AFI), and version change statistics categorized by interval (e.g., 5, 15, 30 minutes). It includes detailed breakdowns by traffic type and source.

The show command output categorizes changes by neighbor IP addresses, VRFs, and AFIs, such as IPv4 Unicast and VPNv4 Unicast, facilitating the analysis of traffic types and identification of version number fluctuation origins.

Step 2 Use the **show bgp sessions version-rate live** command to view the real-time BGP session version-rate statistics, capturing changes within the most recent 5-minute interval across all BGP neighbors and address families.

Example:

```
Router# show bgp sessions version-rate live
Thu Nov  2 11:40:48.314 IST
Interval definition(s):
  Interval   Duration (min)  Start time            End time
  Live                   5  Nov  2 11:37:22.029   Nov  2 11:40:48.610

Neighbor   VRF       AFI                   Total        Live           Spk
                                                    Total Reach Unreach
------------------------------------------------------------------------------
10.10.10.1    default   All                    5      0     0     0        0
10.10.10.1              IPv4 Unicast           5      0     0     0
192.168.0.5   default   All                  606      0     0     0        0
192.168.0.5             IPv4 Unicast          63      0     0     0
192.168.0.5             VPNv4 Unicast        240      0     0     0
192.168.0.5             IPv6 Labeled-unicast  63      0     0     0
192.168.0.5             VPNv6 Unicast        240      0     0     0
192.168.0.5             RT Constraint          0      0     0     0
10:10:10::1    default  All                    5      0     0     0        0
10:10:10::1             IPv6 Unicast           5      0     0     0
10.0.1.1       1        All                    5      0     0     0        0
10.0.1.1                IPv4 Unicast           5      0     0     0
10:0:1::1      1        All                    5      0     0     0        0

/* The output was too wide, so it was segmented; the below segment continues from above.
 */
AS  InQ  OutQ  BRState    NSRState
-------------------------------------
200  0   0    Established  NSRReady

100  0   0    Established  NSRReady

200  0   0    Established  NSRReady


200  0   0    Established  NSRReady

200  0   0    Established  NSRReady
```

The command reports current version changes and session metrics, including total version changes, reachability, speaker ID, AS number, queue sizes, and session states.

This provides a concise snapshot of BGP session activity and health.

Step 3 Use the **show bgp sessions version-rate brief** command to view concise information only for the "Live" interval, which is typically the most recent 5-minute window.

Example:

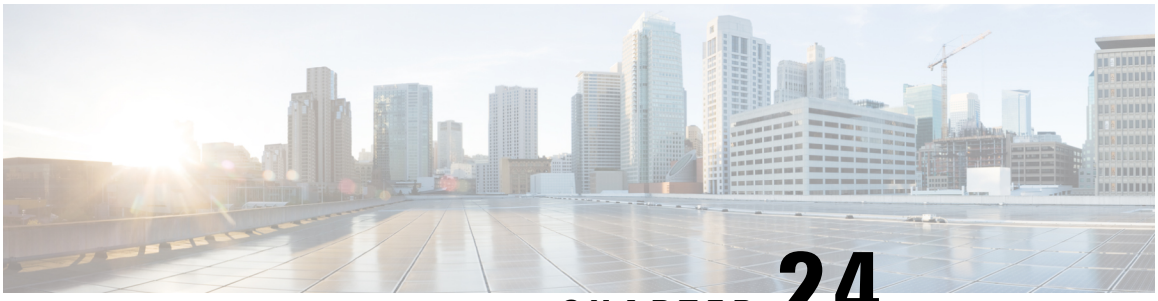
```
Router# show bgp sessions version-rate brief live

Thu Nov  2 11:40:55.743 IST
Interval definition(s):
  Interval   Duration (min)  Start time            End time
  Live                   5  Nov  2 11:37:22.029   Nov  2 11:40:56.059

Neighbor      VRF         Spk    AS   InQ  OutQ  NBRState     NSRState  Total       Live
10.10.10.1    default      0   200     0     0  Established  NSRReady      5          0
192.168.0.5   default      0   100     0     0  Established  NSRReady    606          0
10:10:10::1   default      0   200     0     0  Established  NSRReady      5          0
10.0.1.1        1          0   200     0     0  Established  NSRReady      5          0
10:0:1::1       1          0   200     0
```

The output focuses exclusively on real-time or near real-time BGP version rate statistics, showing only the "Live" interval column next to the "Total," without historical interval details.

You can monitor and analyze BGP version changes in detail using both historical intervals and real-time snapshots, categorized by neighbor, VRF, and AFI. This facilitates early detection of route instability, excessive churn, or underlying network issues.

What to do next

Review the output for any unexpected spikes or patterns in version changes. Investigate specific neighbors or intervals as needed for further troubleshooting or optimization.



CHAPTER 24

Graceful Maintenance

This chapter provides a comprehensive guide to BGP graceful maintenance and failover features. It covers mechanisms like BGP-RIB feedback, extended route retention, nonstop routing, and fast fallover to ensure network stability and minimize traffic loss during various events. Additionally, it details BGP persistence (LLGR) and graceful maintenance for controlled traffic shifts during planned outages and failures.

BGP-RIB feedback mechanisms for update generation

The BGP-RIB feedback mechanisms for update generation are BGP route control mechanisms that

- ensure routes are installed locally before advertising them to neighbors
- track which route versions the forwarding information base (FIB) has consumed, and
- send updates only for routes confirmed installed in the FIB to prevent premature advertisements.

These mechanisms help you avoid traffic loss caused by premature route advertisements after events like router reloads, line card online insertion and removal (LC OIR), or link flaps when alternate paths become available

You configure BGP to wait for routing information base (RIB) feedback before sending updates by using the **update wait-install** command in the `router address-family IPv4` or `router address-family VPNv4` configuration mode. This command ensures that BGP sends updates only after routes are confirmed installed in the forwarding information base (FIB), preventing premature route advertisements.

You can verify this configuration using the following commands:

- **show bgp**
- **show bgp neighbors**

- **show bgp process performance-statistics**

This configuration helps you avoid traffic loss caused by premature route advertisements after events such as router reloads, line card online insertion and removal (LC OIR), or link flaps when alternate paths become available.

Guidelines for BGP-RIB feedback mechanisms

To prevent traffic loss and ensure reliable routing, always advertise BGP routes only after they are confirmed as installed in the Forwarding Information Base (FIB) via the BGP-RIB feedback mechanism.

- ensure that BGP installs routes in the Routing Information Base (RIB) and waits for feedback from the RIB about installation in the FIB.
- confirm that the RIB tracks which route versions are in the FIB using the BCDL feedback mechanism.
- send BGP update messages only for routes confirmed as installed in the FIB, preventing premature advertisements that could cause packet loss or blackholing.

Configure BGP to wait for RIB feedback before sending updates

Enable BGP to delay advertising updates until routes are confirmed as installed in the FIB, preventing premature updates and possible traffic loss.

Use this configuration to enhance BGP routing reliability by ensuring updates are sent only after successful RIB-to-FIB installation confirmation.

Before you begin

- Verify you have administrator access to the router.
- Identify the AS number and desired address family (e.g., IPv4 unicast, VPNv4).

Follow these steps to configure BGP to wait for RIB feedback:

Procedure

Step 1 Enter router configuration mode for the desired address family.

Example:

```
Router# configure
Router(config)# router bgp 1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# update wait-install
```

Step 2 Save the configuration.

Example:

```
Router#(config-bgp-af)# commit
```

Step 3 Use the **show bgp process** command to view the delay of the BGP process update since the last router reload.

Example:

```
Router# show bgp process
Wed Aug 24 00:40:48.649 PDT

BGP Process Information:
BGP is operating in STANDALONE mode
Autonomous System number format: ASPLAIN
Autonomous System: 100
Router ID: 192.168.0.2 (manually configured)
Default Cluster ID: 192.168.0.2
Active Cluster IDs:  192.168.0.2
------------------------------
------------------------------
Update wait-install enabled:
  ack request 2, ack rcvd 2, slow ack 0
  startup delay 10 secs
```

BGP on the router now delays advertising route updates until RIB confirms that routes are installed in the FIB, ensuring reliable route propagation and preventing traffic loss.

What to do next

Monitor BGP updates and verify network stability after applying the configuration.

BGP extended route retention

BGP extended route retention is a routing feature that

- applies a route retention policy to modify route attributes when a BGP peer fails
- modifies route attributes in addition to changes caused by the neighbor's inbound policy, and
- enables the use of route retention policy instead of Long-Lived Graceful Restart (LLGR) when the BGP hold timer expires or when the BGP session fails to reestablish within the configured graceful restart time

This feature helps maintain route stability and control during BGP peer failures by retaining routes with modified attributes until the session is restored.

Table 58: Feature History Table

Feature Name	Release Name	Description
BGP Extended Route Retention	Release 7.3.3	This feature allows you to maintain stale routing information from a failed BGP peer for longer periods of time than that is configured in the Graceful Restart atribute. However, this feature ensures that the BGP neighbor considers the stale routes as new routes.

Recommendations for using extended route retention

Adhere to the following principles to ensure proper operation and compatibility when using BGP Extended Route Retention:

- Ensure that your BGP neighbor supports graceful restart functionality.
- Apply graceful restart functionality when a BGP neighbor fails, and maintain it until the graceful restart timer expires.
- Start the Extended Route Retention feature only after the graceful restart timer expires.
- Configure soft-reconfiguration inbound as a mandatory setting; apply inbound policy if required.
- Activate Extended Route Retention exclusively when the BGP peer goes down, specifically after the hold-down timer expires.
- Do not treat routes as stale or retain them for any other triggers such as timer expiry; in such cases, purge the routes.
- Use Extended Route Retention only with the following address-family modes: IPv4 and IPv6 unicast, IPv4 and IPv4 labelled unicast.
- Do not configure both LLGR and Extended Route Retention on the same neighbor.
- Do not send the capability attribute when Extended Route Retention is configured.

Configure route policies and apply them to a BGP neighbor

Define route policies with specific community and local preference settings, then apply these policies to a BGP neighbor to control routing behavior, including route retention and inbound policy processing.

Use this task when you need to define route policies with specific community and local preference settings and apply them to a BGP neighbor, including route retention and inbound policies.

Before you begin

- Ensure you have the necessary privileges to configure BGP and route policies on the router.
- Identify the names for the route-policies and communities.

Procedure

Step 1 Create route policies.

Example:

Create the route policy RRP_comm_no_export_local_pref_2500.

```
Router(config)# route-policy RRP_comm_no_export_local_pref_2500
Router(config-rpl)# set community RRP_comm_no_export additive
Router(config-rpl)# set local-preference 2500
Router(config-rpl)# end-policy
Router(config-rpl)# exit
```

Create the route policy comm_number_local_pref.

```
Router(config)# route-policy comm_number_local_pref
Router(config-rpl)# set community comm_number
Router(config-rpl)# set local-preference 10000
Router(config-rpl)# end-policy
Router(config-rpl)# exit
```

Step 2 Apply the route policies to a BGP neighbor.

Example:

Enter BGP router configuration mode for AS 140:

```
Router(config)# router bgp 140
```

Configure the neighbor with IP address 10.1.1.1.

```
Router(config-bgp)# neighbor 10.1.1.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# update-source Loopback 0
```

Enter the IPv4 unicast address family.

```
Router(config-bgp-nbr)# address-family ipv4 unicast
```

Apply the inbound route policy.

```
Router(config-bgp-nbr-af)# route-policy RRP_comm_no_export_local_pref_2500 retention retention-time
 2340
```

Enable soft reconfiguration to view the peer's adj-rib-in table.

```
Router(config-bgp-nbr-af)# soft-reconfiguration inbound always
```

The route policies are now successfully configured and applied to the BGP neighbor. This setup controls community attributes, local preference values, route retention duration, and inbound policy processing.

Step 3 Use the **show bgp neighbor** command to verify the configured route-retention policy using show bgp neighbor.

Example:

```
Router# show bgp neighbor 10.1.1.1
Fri Oct 22 04:52:44.972 PDT

BGP neighbor is 10.1.1.1
 Remote AS 1, local AS 1, internal link
 Remote router ID 10.1.1.1
  BGP state = Established, up for 00:03:03
…
 For Address Family: IPv4 Unicast
  BGP neighbor version 16172
Policy for incoming advertisements is comm_number_local_pref
  Policy for Retention is RRP_comm_no_export_local_pref_2500
Configured route retention policy stale timer for routes is 2340 seconds
….
```

The `show bgp neighbor 10.1.1.1` output confirms:

- BGP neighbor 10.1.1.1 is established and up for 3 minutes.
- Remote AS is 1; local AS is 1 (internal link).
- Inbound policy: `comm_number_local_pref`

  .
- Route-retention policy: `RRP_comm_no_export_local_pref_2500`

.

- Stale timer set to 2340 seconds.

This verifies the route policies and retention timer are correctly applied and active.

Step 4 Use the **show bgp ipv4 unicast** command to verify the presence and status of stale routes using show bgp ipv4 unicast.

Example:

```
Router# show bgp ipv4 unicast 181.1.1.0/24
Fri Oct 22 04:56:15.906 PDT
….
Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    3.3.3.3
  100, (Received from a RR-client), (long-lived/route-retention policy stale)
    192.1.2.1 (metric 10) from 192.1.2.1 (10.1.1.1)
      Origin IGP, metric 1221, localpref 2500, valid, internal, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 16243
      Community: 1:100 no-export
```

This output confirms that the route is marked as stale due to the route-retention policy. It verifeies that the route's attributes such as origin, metric, local preference, and community, and shows the route is valid, internal, and selected as the best path. The output also confirms the route is advertised to peers and includes path identifiers for tracking. This demonstrates the route-retention policy is effectively maintaining stale routes for network stability.

Route policies with specific attributes are successfully applied to the BGP neighbor, providing route retention and policy enforcement.

BGP nonstop routing with stateful switchovers

A BGP nonstop routing is a network protocol feature that

- enables all BGP peerings to maintain BGP states, and
- ensures continuous packet forwarding without interruption visible to peer routers by maintaining protocol sessions and routing states across process restarts and switchovers.

Guidelines for BGP nonstop routing with stateful switchovers

- Configure the **nsr process-failures switchover** command to maintain nonstop routing (NSR) during BGP or TCP process crashes.
- Without this configuration, BGP neighbor sessions will flap, causing network instability.
- NSR does not prevent session flapping during BGP or TCP process restarts; expect neighbor sessions to flap in such cases.
- Additional measures beyond NSR are required to manage session flapping caused by process restarts.
- This command is mandatory to ensure network stability during process failures.

How active and standby route processors work during switchovers and failures

Summary

This process explains how active and standby route processors, along with BGP and TCP processes, maintain continuous network operation during route processor switchovers or failures. The key components involved in the process are:

- Active route processor: Manages routing and session handling under normal conditions.
- Standby route processor: Synchronizes state with the active processor and takes over when needed.
- BGP process: Maintains Border Gateway Protocol sessions and routing information.
- TCP process: Maintains TCP connections supporting BGP sessions.

Workflow

The process involves the following stages:

1. Switchover or failure detection: The system detects when the active route processor fails or switches over.
2. State synchronization: Synchronization points ensure consistent internal state between active and standby BGP and TCP processes.
3. Session migration: TCP connections and BGP sessions transparently migrate to the standby processor, which becomes active.
4. Continuous forwarding: Packet forwarding continues uninterrupted without requiring peer routers to refresh protocol states or upgrade software.

Result

This process ensures uninterrupted packet forwarding and BGP session continuity during route processor switchovers or failures, maintaining network stability without manual intervention.

Capabilities and limitations of BGP nonstop routing

NSR maintains active BGP and TCP sessions during route processor switchovers, process crashes, and system upgrades by transparently failing over to the standby route processor without interrupting packet forwarding or causing session flaps. This ensures continuous network operation and enhances stability during critical events.

BGP nonstop routing capabilities

- NSR-related alarms and notifications
- Separate tracking of configured and operational NSR states
- NSR statistics collection
- NSR statistics display via `show` commands
- XML schema support
- Auditing mechanisms for active/standby state synchronization

- CLI commands for NSR enablement and disablement

Events triggering NSR

- Route processor switchovers
- BGP or TCP process crashes or failures
- Restart of `l2vpn_mgr` causing state flapping (no traffic loss)
- In-Service System Upgrades (ISSUs) involving stateful switchover (SSO)

Enabled capabilities

- Transparent migration of TCP connections and BGP sessions to standby route processor preserving protocol state without peer refresh

  NSR-related alarms and notifications
- Separate tracking of configured and operational NSR states
- Collection and display of NSR statistics via show commands
- XML schema support for NSR data
- Auditing mechanisms for state synchronization verification
- Support for up to 5000 NSR sessions
- No requirement for software upgrades or NSR support on peer routers

Limitations

- NSR does not prevent session flapping if the BGP or TCP process restarts; expect neighbor sessions to flap in such cases.
- When the `l2vpn_mgr` process restarts, the NSR client (`te-control`) may flap between Ready and Not Ready states, which is expected and causes no traffic loss.

Configuration requirements

- Configure the **nsr process-failures switchover** command to maintain NSR during BGP or TCP process crashes or failures.
- Without this command, BGP neighbor sessions will flap during process crashes.
- NSR does not prevent session flapping during BGP or TCP process restarts; expect neighbor sessions to flap in such cases.
- Additional measures beyond NSR may be required to manage session flapping caused by process restarts.

Additional operational notes

- During route processor switchovers and In-Service System Upgrades (ISSUs), NSR is achieved by stateful switchover (SSO) of both TCP and BGP.

- NSR does not require software upgrades on peer routers, nor do peers need to support NSR.
- When a route processor switchover occurs due to a fault, TCP connections and BGP sessions migrate transparently to the standby RP, preserving protocol state without requiring peer refresh.
- Events like soft reconfiguration and policy changes can alter BGP internal states; synchronization points called post-its keep active and standby BGP processes aligned.

Enable BGP NSR

Configure BGP NSR for BGP to enhance process resiliency.

NSR allows BGP sessions to remain up during process restarts. Enable NSR to improve network availability

Before you begin

Ensure you are in privileged EXEC mode on the router.

Save your current configuration.

Procedure

Enable NSR.

Example:

```
Router# router bgp 120
Routing(config-bgp)# nsr
```

The router enables BGP NSR as configured.

Disable BGP NSR

To remove non-stop routing for BGP or troubleshoot related issues, disable NSR for BGP.

NSR allows BGP sessions to remain up during process restarts. Disable NSR if troubleshooting or removing the feature.

Before you begin

Ensure you are in privileged EXEC mode on the router.

Save your current configuration.

Procedure

Disable NSR.

Example:

```
Router# router bgp 120
Routing(config-bgp)# no nsr
```

The router disables BGP NSR as configured.

BGP fast external fallover

BGP fast external fallover is a routing feature that

- automatically resets all BGP sessions of directly adjacent external peers when a link goes down
- ensures immediate response to link failures affecting eBGP connections, and
- can be disabled or re-enabled to favor either session stability or rapid convergence.

Guidelines for BGP fast external failover

To prevent eBGP session flapping when your node reaches 3500 eBGP sessions with BGP timer values set to 10 and 30, increase the packet rate. Use the **lpts pifib hardware police location location-id** command to raise the packet rate and support more than 3500 eBGP sessions.

Applies when managing high numbers of eBGP sessions on a node.

Increasing the packet rate prevents session instability caused by timer settings and session volume.

Ensures stable BGP session management under heavy session loads.

Configure the packet rate using the command **lpts pifib hardware police location** command.

Configure the eBGP session packet rate

This task instructs you on how to increase the packet rate for BGP traffic, which helps support a higher number of eBGP sessions and prevent session flapping in large-scale deployments.

Before you begin

Ensure you have access to the router's global configuration mode and the necessary permissions to modify LPTS PIFIB hardware policing settings.

Procedure

To increase the packet rate for BGP traffic and support a higher number of eBGP sessions, configure the LPTS PIFIB hardware policing settings as follows

Example:

```
Router# configure
Router(config)# lpts pifib hardware police location 0/2/CPU0
Router(config-pifib-policer-per-node)# flow bgp configured rate 4000
```

```
Router(config-pifib-policer-per-node)# flow bgp known rate 4000
Router(config-pifib-policer-per-node)# flow bgp default rate 4000
```

This configuration helps prevent session flapping in large-scale deployments by allowing the router to handle increased BGP packet rates efficiently.

BGP fast fallover

BGP fast fallover is a routing feature that

- quickly removes routes learned from directly connected iBGP or eBGP neighbors when an IP interface fails
- accelerates the network convergence process by preventing the propagation of stale routes, and
- eliminates the need to wait for the hold timer to expire when a directly attached interface fails.

When an interface attached to a directly connected BGP neighbor fails, routes learned from that neighbor typically persist until the hold timer expires. This lag can lead to slow network convergence and potential network instability. BGP Fast Fallover addresses this by ensuring routes are removed immediately. You can also use the `nexthop trigger-delay` command to quickly remove BGP routes of a failing neighbor, provided that the neighbor's BGP session endpoint is the same as the route's next hop.

Table 59: Feature History Table

Feature Name	Release Information	Feature Description
BGP Fast Fallover	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on the Cisco 8712-MOD-M routers.
BGP Fast Fallover	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 88-LC1-36EH+A8:B12 • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8212-48FH-M • 8711-32FH-M

BGP Fast Fallover	Release 24.2.11	You can now terminate the external BGP sessions to an adjacent peer when the link to that peer goes down, without waiting for the hold timer to expire. With this feature you can enable fast fallover mechanism on a specific BGP neighbor even if **bgp fast-external-failover disable** command is globally configured. This feature enables quicker failure detection, and allows other recovery mechanisms to reroute the traffic quickly, thus resulting in faster convergence. The feature introduces these changes: CLI: • **fast-fallover** YANG Data Model: • Cisco-IOS-XR-um-router-bgp-cfg.yang (see GitHub, YANG Data Models Navigator)

Guidelines for BGP fast fallover

To ensure proper operation and network stability when using BGP fast fallover, follow these guidelines:

- Apply fast fallover only to directly connected BGP neighbors. A directly connected neighbor is one that is either one hop away or has an IP address within the same subnet. Do not apply fast fallover to neighbors connected through loopback interfaces, even if they are one hop away.
- Maintain established BGP sessions with neighbors that are not directly connected until a triggering event, such as hold timer expiration, causes the session to go down.
- If an interface fails before fast fallover activates, manually clear the BGP neighbor session if necessary, as the BGP session does not automatically go down.
- Allow the regular BGP session establishment process to proceed unchanged when an interface recovers from failure.

Follow these recommendations to maintain predictable BGP session behavior and ensure network stability when using the fast fallover feature.

Configure BGP fast fallover

This task describes how to enable and verify the BGP fast fallover feature for neighbors.

Before you begin

- Verify you have appropriate access to the router's configuration mode.
- Confirm that you understand the network topology, especially which BGP neighbors are directly connected.
- Ensure you have the necessary permissions to modify BGP neighbor configurations.

Procedure

Step 1 Enable fast fallover.

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# neighbor 209.165.201.0
Router(config-bgp-nbr)# fast-fallover
```

- By default, fast fallover is enabled for eBGP neighbors and disabled for iBGP neighbors.
- If the `bgp fast-external-fallover disable` command is configured globally or in VRF mode, fast fallover is disabled for eBGP neighbors but can be overridden per neighbor using the `fast-fallover` command.
- To stop the fast fallover setting from being inherited from a higher-level neighbor or session group, use the `fast-fallover inheritance-disable` command.

Step 2 Verify fast fallover configuration.

Example:

Use the **show bgp neighbors ip-address** or **show run router bgp as-number neighbor ip-address** command to check if fast fallover is enabled or inherited.

```
Router# show bgp neighbors 209.165.201.0
BGP neighbor is 209.165.201.0
...
fast-fallover
address-family ipv4 unicast
 use af-group ipv4_unicast_3_ibgp
...
Fast fallover is enabled
Neighbor is directly connected
Neighbor fast-fallover is configured
Neighbor is external and fast-external-fallover is not disabled
...
```

The presence of fast-fallover in the configuration output confirms that the feature is successfully configured for the neighbor.

BGP persistence

BGP persistence mechanism is a routing feature that

- allow you to retain BGP routes learned from a configured neighbor
- keep those routes active even if the BGP session with that neighbor goes down, and
- help you maintain network stability during transient neighbor outages.

BGP persistence, also known as Long-Lived Graceful Restart (LLGR), is a feature that enables a local router to retain BGP routes learned from a configured neighbor, even if the BGP session with that neighbor goes

down. This capability helps maintain network stability by preventing unnecessary route withdrawals during transient neighbor outages.

Key characteristics of BGP Persistence routes include:

- Duration: LLGR can remain in effect for a significantly longer period than standard Graceful Restart (GR).
- Route Preference: LLGR stale routes are assigned the lowest preference during the BGP bestpath computation, ensuring that fresh, active routes are always preferred.
- Advertisement: If an LLGR stale route is selected as the best path, it is advertised with the `LLGR_STALE` community (65535:6) attached. These routes are not advertised to neighbors that do not support LLGR.
- Resilience: LLGR stale routes are not deleted if the forwarding path to the neighbor is detected as down, nor are they deleted if the BGP session to the neighbor experiences multiple flaps, even if the neighbor does not re-advertise the route.
- Exclusion: Any route explicitly tagged with the `NO_LLGR` community (65535:7) will not be retained by the BGP persistence mechanism.

By using BGP persistence, you reduce the impact of transient BGP session outages on your network, which can be especially useful in large or dynamic environments.

Limitations for BGP persistence

The BGP persistence feature is supported only on the following address family identifiers:

- VPNv4 and VPNv6
- RT constraint
- Flowspec (IPv4, IPv6, VPNv4, and VPNv6)
- IPv4 and IPv6 address families

BGP persistence operational flow

Summary

BGP persistence operational flow begins when a BGP session drops or after Graceful Restart, with its capability signaled during session establishment. The local router then retains learned routes as stale. LLGR concludes upon stale timer expiry or receipt of an End-of-RIB marker, at which point any remaining stale routes are deleted.

Workflow

BGP persistence operates through a defined lifecycle as given below:

1. Initiation: LLGR takes effect either immediately upon a BGP session going down (if standard Graceful Restart is not enabled) or after the standard Graceful Restart process concludes.
2. Capability Signaling: The LLGR capability is signaled to a neighbor during the BGP session establishment via the BGP OPEN message, provided it has been configured for that neighbor.

3. Route Retention: Once active, the local router retains learned routes from the neighbor, marking them as "stale" but keeping them in the routing table.
4. Termination: LLGR for a neighbor ends when one of the following conditions is met:
   - The configured LLGR stale timer expires.
   - The neighbor sends an End-of-RIB (Routing Information Base) marker, indicating that it has completed revising and re-advertising its routes.
5. Stale Route Deletion: Upon LLGR termination, any routes from that neighbor that are still marked as stale are deleted from the routing table.

Configure BGP persistence

Use this task to enable BGP Persistence (Long-Lived Graceful Restart) for a BGP neighbor, allowing the router to retain routes during session outages.

BGP Persistence, also known as Long-Lived Graceful Restart (LLGR), ensures network stability by keeping learned routes active even when a neighbor session is temporarily down. This configuration defines how long stale routes are retained.

Before you begin

Ensure that you meet the following requirements:

- You must have a basic BGP configuration already in place, including the BGP process and the neighbor definition.
- You must know the remote Autonomous System (AS) number and the IP address of the BGP neighbor for which you are configuring LLGR.
- Understand the implications of **graceful-restart stalepath-time** if configured, as LLGR takes effect after standard Graceful Restart concludes.

To configure BGP Persistence for a neighbor, perform the following steps:

Procedure

Step 1 Configure BGP on the router, and the BGP neighbor, and its basic parameters.

Example:

```
Router(config)# router bgp 100
Router(config-router)# neighbor 10.3.3.3
Router(config-router-neighbor)# remote-as 30813
Router(config-router-neighbor)# update-source Loopback0
Router(config-router-neighbor)# graceful-restart stalepath-time 150
Router(config-router-neighbor)#
```

Step 2 Enter address family configuration mode for VPNv4 unicast.

Example:

```
Router(config-router-neighbor)# address-family vpnv4 unicast
```

Step 3 Enable long-lived graceful restart capability for the VPNv4 address family and specify the LLGR stale time for sending and accepting VPNv4 routes.

Example:

```
Router(config-router-af)# long-lived-graceful-restart capable
Router(config-router-af)# long-lived-graceful-restart stale-time send 16777215 accept 16777215
Router(config-router-af)# exit
```

Step 4 Enter address family configuration mode for VPNv6 unicast.

Example:

```
Router(config-router-neighbor)# address-family vpnv6 unicast
```

Step 5 Enable long-lived graceful restart capability for the VPNv6 address family and specify the LLGR stale time for sending and accepting VPNv6 routes.

Example:

```
Router(config-router-af)# long-lived-graceful-restart capable
Router(config-router-af)# long-lived-graceful-restart stale-time send 16777215 accept 16777215
Router(config-router-af)# exit
```

Flexible BGP persistence

Flexible BGP persistence is a routing feature that

- enhances network stability and resilience
- enables Long-Lived Graceful Restart (LLGR) with flexible stale time management, and
- allows controlled route distribution within the Autonomous System (AS).

This feature provides the flexibility to advertise LLGR stale routes to both LLGR-capable and non-LLGR-capable neighbors, ensuring continuous route availability during planned or unplanned restarts. It simplifies configuration by removing the need for manual timeout settings and enforces controlled route propagation by attaching specific BGP community attributes.

Table 60: Feature History Table

Feature Name	Release Name	Description
Flexible BGP Persistence	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]). This feature is supported on Cisco 8712-MOD-M routers.

Feature Name	Release Name	Description
Flexible BGP Persistence	Release 24.3.1	

Feature Name	Release Name	Description
		Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) Now you can ensure continuous connectivity by allowing non-Long Lived Graceful Restart (LLGR) eBGP neighbors to use LLGR stale routes, allowing for LLGR capability to be enabled and advertised without having to explicitly configure a timeout value, and gain greater flexibility in route management by advertising stale routes to non-LLGR peers through the NO_EXPORT community. This is an enhancement to the existing BGP Persistence feature. The feature introduces these changes: CLI: • The **default**, **any**, and **advertise-internal-only** keywords are added to the long-lived-graceful-restart command. • The fields **ault advertised long-lived stale time**, and **Long-lived Graceful Restart Stale Time Accept Any** are added to the show output of the show bgp command. YANG Data Model: • `Cisco-IOS-XR-ipv4-bgp-cfg` (see GitHub, YANG Data Models Navigator) *This feature is supported on: • 88-LC1-36EH

Feature Name	Release Name	Description
		• 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8212-48FH-M • 8711-32FH-M

Benefits of flexible BGP persistence

Flexible BGP Persistence offers the following benefits:

- Simplified configuration: Enables LLGR without manual timeout configuration using the **long-lived-graceful-restart stale-time send default accept any** command, which advertises a default stale time and accepts peer-specified stale times.
- Enhanced network resilience: Allows LLGR stale routes to be advertised to non-LLGR eBGP neighbors, improving overall network robustness.
- Enhanced network stability: Attaches the `NO_EXPORT` community and sets local preference to 0 for LLGR routes advertised to internal neighbors without LLGR capability, preventing stale routes from propagating beyond the local AS and ensuring they are not preferred over other routes.

Configure LLGR advertisement and activation with default and peer-time values

Enable and advertise Long-Lived Graceful Restart (LLGR) capability using default and peer-specified stale time values.

Before you begin

Ensure you have router BGP configuration access.

Procedure

Step 1 Enable and advertise LLGR capability with default and peer stale times.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 10.1.1.1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# long-lived-graceful-restart stale-time send default accept any
```

This command advertises LLGR capability with a default stale time of 172,800 seconds (2 days) and accepts any stale time value set by the peer.

Step 2 Use the **show bgp neighbor** to verify the LLGR configuration and stale time settings.

Example:

```
Router(config)# show bgp neighbor 192.0.2.254
...
AF-dependent capabilities:
```

```
...
Long-lived Graceful Restart Stale Time Send Default is ON
Default advertised long-lived stale time is 172800 seconds
Long-lived Graceful Restart Stale Time Accept Any is ON
....
```

These output lines indicate these settings.

- Long-lived Graceful Restart Stale Time Send Default is ON.
- Default advertised long-lived stale time is 172800 seconds.
- Long-lived Graceful Restart Stale Time Accept Any is ON.

LLGR capability is enabled on the BGP neighbor with default and peer-specified stale times successfully advertised and verified.

Enable LLGR capability and advertise it only to iBGP peers

Enable long lived graceful restart (LLGR) capability and restrict advertisement to iBGP peers.

Before you begin

Confirm BGP neighbor configuration.

Procedure

Step 1 Enable LLGR capability and advertise it only to iBGP peers:

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 10.1.1.1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# long-lived-graceful-restart capable advertise-internal-only
```

This step sets the local preference to 0 and attaches the `llgr-stale` and `no-export` community attributes to limit route propagation within the local AS.

Step 2 Use the **show bgp** command to that the route is not preferred for outbound traffic within the AS, and that it will not be advertised to external BGP peers, thereby limiting its propagation to within the local AS.

Example:

```
Router# show bgp 10.1.1.1
Path #32: Received by speaker 0
…
192.0.2.254 (metric 30) from 10.1.1.1 (192.0.2.254)
Origin IGP, localpref 0, valid, internal, add-path
Received Path ID 40, Local Path ID 9, version 14321
Community: llgr-stale no-export
Originator: 192.0.2.254, Cluster list: 10.1.1.1
```

These output lines indicate these settings.

- Local preference 0

- Community attributes: llgr-stale no-export

BGP graceful maintenance

BGP graceful maintenance is a routing feature that

- allows routers or links to remain in service while the network reroutes traffic to alternative paths,
- minimizes traffic loss during convergence by enabling the network to find alternate routes before taking a router or link out of service, and
- supports both shutdown and startup scenarios to reduce traffic disruption.

This feature is especially useful in networks with long convergence times caused by factors such as large routing tables or route reflectors.

Restrictions for BGP graceful maintenance

These restrictions apply to BGP graceful maintenance:

- Routers configured to send the GSHUT community attribute require other routers to interpret it via matching routing policies and setting lower preferences.
- The LOCAL_PREF attribute is not sent to another AS, so it cannot be used on eBGP links, except between member-ASs of an AS confederation.
- Alternate routes must exist in the network; otherwise, advertising a lower preference has no effect (e.g., singly-homed customer routers without alternate routes).

Graceful maintenance operation

When you activate BGP Graceful Maintenance, you ensure a controlled traffic shift by signaling reduced route preference to neighboring routers, prompting them to select alternative paths.

- Signaling Reduced Preference: You can signal reduced route preference using these methods:
  - Add GSHUT community: This method allows remote routers the flexibility to define their own preference based on a policy match.
  - Reduce LOCAL_PREF value: This method is effective for internal BGP neighbors, especially if remote routers do not process the GSHUT community.
  - Prepend AS Path: This method works for both internal and external BGP neighbors, particularly when remote routers do not recognize the GSHUT community.
- Impact on BGP Connections: When you activate Graceful Maintenance on a BGP connection, two key operations occur:

- Re-advertisement of received routes: All routes received from the connection are re-advertised to other neighbors with a lower preference. This applies only to routes that were originally advertised to those neighbors.
- Re-advertisement of advertised routes: All routes previously advertised *to* the connection are re-advertised with a lower preference.

- Internal Tagging for Received Routes: To facilitate the re-advertisement of received routes, we internally tag them with a `graceful-shut` attribute. This attribute is local to the router and is not advertised via BGP. You can view this attribute using the **show bgp** command. This differs from the `GSHUT` community, which is advertised by BGP.
- Route Selection Preference: All routes possessing the `graceful-shut` attribute receive the lowest preference during route selection. Any new route updates exchanged on a BGP session under Graceful Maintenance are processed with this same preference adjustment.

Guidelines for inter-autonomous system usage

Advertise a lower preference to another autonomous system (AS) in the public Internet only when necessary. Unnecessary advertisement may cause excessive routing updates in remote networks, which can be undesirable.

Key points:

- Avoid advertising lower preference unless required to prevent unnecessary routing advertisements.
- Configure the router with the `send-community-gshut-ebgp` setting under the neighbor address family to originate the GSHUT community to the eBGP neighbor.
- Be aware that this setting affects only the GSHUT community you add; it does not alter the GSHUT community on routes you receive.

Best practices for handling the graceful-shut attribute in BGP

- Assign the lowest preference to any BGP route tagged with the graceful-shut attribute during route selection to maintain correct routing behavior.
- Remember that the graceful-shut attribute is internal and is not advertised to external BGP peers.
- Do not confuse the graceful-shut attribute with the GSHUT community, which is advertised externally by BGP.
- During Graceful Maintenance, treat any new route updates received or sent on a BGP session in the same way.

This approach ensures proper route handling and avoids inadvertent routing preferences during network maintenance.

Requirement: Verify network convergence before performing a graceful maintenance shutdown

Always confirm full network convergence before shutting down a router or link after activating graceful maintenance. This step is essential to prevent traffic loss and service disruption.

Premature shutdown before convergence can cause severe operational hazards and substantial traffic loss. Network-wide convergence may take seconds to over an hour and cannot be determined solely by the local router state.

By verifying convergence, you minimize the risk of traffic blackholing and ensure a smooth, uninterrupted transition during maintenance.

Apply these instructions every time you perform graceful maintenance and intend to shut down a router or link. Skipping this step can lead to critical service impacts.

To determine full convergence:

- Monitor BGP messaging queues and traffic flow.
- Use the **show bgp <vrf> <afi> <safi> summary** commands to confirm both **InQ** and **OutQ** are zero) for all neighbors.
- Ensure traffic is no longer sent to the router being maintained.

Configure graceful maintenance on a BGP router for all neighbors

Advertise routes with the GSHUT community to enable graceful service maintenance across all BGP neighbors through a single configuration.

Before you begin

- Verify you have the required access permissions and credentials to configure the BGP router.
- Confirm that the router is operational and that BGP sessions with neighbors are established.

Procedure

Enter BGP router configuration mode and configure graceful maintenance for all neighbors.

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# graceful-maintenance activate all-neighbors
```

What to do next

- After activating Graceful Maintenance, wait for all routes to be sent and for neighboring routers to redirect traffic away from the router or link under maintenance. Once traffic is redirected, it is safe to take the router or link out of service.

- To monitor progress, use the `show bgp summary` command and check the `OutQ` value for neighbors. When `OutQ` reaches 0, no more updates remain to be sent.

Configure graceful maintenance on a neighbor

Enable the router to announce routes with graceful maintenance attributes, including the GSHUT community, to a neighbor.

Before you begin

Verify the IP address of the BGP neighbor is known and reachable.

Procedure

Enable graceful maintenance on BGP neighbor.

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbrgrp)# graceful-maintenance activate
Router(config-bgp)# commit
```

Activating graceful maintenance on a single neighbor enables coordinated route advertisement with GSHUT community attributes for the specific neighbor in the group, facilitating efficient maintenance operations.

Graceful maintenance is activated for the specified neighbor, enabling coordinated route advertisement with the GSHUT community attribute for maintenance operations.

Configure the router to reduce route preference for graceful maintenance

Allow alternate BGP routes to take over before removing a link or router by lowering route preference.

Use this procedure to safely enable BGP graceful maintenance without routing disruption. Lowering the route preference (local-preference) ensures traffic shifts to alternative paths while maintaining service continuity.

Before you begin

- Ensure alternate network paths are available to receive redirected traffic.
- Prepare the route policy that matches the GSHUT community to set the lower local preference value.

Follow the steps to reduce the route preference on the router :

Procedure

Step 1 Configure the BGP neighbor for graceful maintenance with a lower local preference:

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# graceful-maintenance local-preference 4
```

Step 2 Define a route policy that matches the GSHUT community and sets the local preference to 0:

Example:

```
Router(config)# route-policy gshut
Router(config-rpl)# if community matches-any gshut then
Router(config-rpl)# set local-preference 0
Router(config-rpl)# endif
Router(config-rpl)# pass
Router(config-rpl)# end-policy
```

Step 3 Apply the route policy inbound on the configured BGP neighbor:

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy gshut in
```

- Attributes for graceful maintenance are added to a route update message after applying an outbound policy.
- Routes received from a GSHUT neighbor are marked with a GSHUT attribute to distinguish them from routes received with the GSHUT community.
- When a neighbor exits maintenance, its GSHUT attribute is removed, but the community tag remains.
- The GSHUT attribute is internal and used only for path selection; it is not sent in BGP messages.

The router advertises routes with lower preference, allowing alternate routes to be selected for traffic before you take down a link or the router.

Configure BGP inbound route policy for GSHUT community

Configure route policy for graceful maintenance by lowering the preference for GSHUT community.

This configuration defines and applies a BGP route policy that sets the `local-preference` to `0` for incoming routes carrying the `gshut` BGP community.

Before you begin

Ensure alternate paths exist: The graceful maintenance feature relies on traffic having other routes to take; without them, this configuration is ineffective.

Ensure you have a backup of the current configuration: Always save a copy of your router's configuration before making any changes to allow for easy rollback if needed.

Follow these steps to configure a BGP inbound route policy that lowers the preference of routes carrying the GSHUT community, and facilitating graceful service maintenance.

Procedure

Configure route policy matching GSHUT community to lower route preference.

Example:

```
Router(config)# route-policy gshut
Router(config-rpl)# if community matches-any gshut then
Router(config-rpl-if)# set local-preference 0
Router(config-rpl-if)# endif
Router(config-rpl-if)# pass
Router(config-rpl-if)# end-policy
Router(config-rpl-if)# exit
Router(config-rpl)# exit
Router(config)# router bgp 120
Router(config-bgp)# neighbor 10.0.0.3
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr)# route-policy gshut in
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config)# exit
```

Verify BGP graceful maintenance activation and attributes

Ensure BGP graceful maintenance is active and correctly configured to allow for seamless routing transitions during planned maintenance.

Use this task to verify that your BGP routers are properly advertising the graceful-shutdown community and associated attributes, preventing network disruptions during maintenance events.

Before you begin

Ensure you have access to the router command.

Confirm you have privileges to run show commands.

Follow these steps to verify BGP graceful maintenance activation and attributes.

Procedure

Step 1 Verify BGP routes with graceful-shutdown community

Example:

```
Router# show bgp 192.0.2.1
...
192.0.2.10 from 192.0.2.10 (198.51.100.1)
Received Label 24000
Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best,
import-candidate
Received Path ID 0, Local Path ID 1, version 4
Community: graceful-shutdown
Originator: 198.51.100.1, Cluster list: 198.51.100.1
```

Review the output for entries showing the `graceful-shutdown` community and related path attributes.

These lines confirm that BGP graceful maintenance is active and that the route is marked accordingly to allow alternate paths to take over before a link or router is taken down:

- Community: graceful-shutdown: This line explicitly shows that the route carries the graceful-shutdown community attribute.
- Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best, import-candidate: These status codes and attributes confirm the route's state, including the presence of the graceful-shut path attribute as part of the route's characteristics.
- Received Path ID 0, Local Path ID 1, version 4: This line provides path identification details relevant to the graceful maintenance feature.

Step 2 Verify the graceful-shutdown community and path attribute.

Example:

```
Router# show bgp community graceful-shutdown
BGP router identifier 198.51.100.1, local AS number 4
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0000000 RD version: 18
BGP main routing table version 18
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
* 10.5.5.5/32 10.10.10.1 88 0 1 ?
Processed 1 prefixes, 1 paths
```

These lines collectively confirm that the graceful maintenance feature is active and that the routes are marked accordingly to allow alternate paths to take over before a link or router is taken down.

- Community: graceful-shutdown: This line explicitly shows that the route carries the graceful-shutdown community attribute, indicating that graceful maintenance is active for this route.
- Network Next Hop Metric LocPrf Weight Path and the following line: `* 10.5.5.5/32 10.10.10.1 88 0 1 ?`: This shows the network prefix and associated attributes, confirming the route is valid and active under graceful maintenance.
- Processed 1 prefixes, 1 paths: This confirms the number of prefixes and paths processed with the graceful-shutdown community.

Step 3 Verify if graceful maintenance is locally active on a BGP neighbor and to view local preference and AS prepends:

Example:

```
Router# show bgp neighbor 192.0.2.1
...
Graceful Maintenance locally active, Local Pref=45, AS prepends=3
...
For Address Family: IPv4 Unicast
...
GSHUT Community attribute sent to this neighbor
...
```

These lines collectively confirm that graceful maintenance is active and that the relevant attributes are applied and communicated to the neighbor:

- Graceful Maintenance locally active, Local Pref=45, AS prepends=3: This line confirms that graceful maintenance is currently active on the neighbor, showing the local preference value and the number of AS path prepends applied.

- For Address Family: IPv4 Unicast: This indicates the address family context for which the graceful maintenance attributes apply.
- GSHUT Community attribute sent to this neighbor: This line shows that the GSHUT community attribute is being sent to the neighbor, which is part of the graceful maintenance signaling.

Step 4 Verify the key attributes of the BGP neighbor that indicates that the graceful maintenance is enabled:

Example:

```
Router# show bgp neighbor 10.12.12.5 configuration

neighbor 10.12.12.5
remote-as 1 []
graceful-maintenance 1 []
gr-maint local-preference 45 []
gr-maint as-prepends 3 []
gr-maint activate []
```

These lines confirm that graceful maintenance is active and configured with the specified local preference and AS path prepends for this neighbour:

- `remote-as 1`
- `gr-maint local-preference 45`
- `gr-maint as-prepends 3`
- `gr-maint activate`

Step 5 List all community set objects and view graceful maintenance feature attributes:

```
Router# show rpl community-set
Listing for all Community Set objects
community-set gshut
graceful-shutdown
end-set
```

These lines display the community sets configured for graceful maintenance on the router.

Step 6 Monitor syslog warning for BGP graceful maintenance activation.

```
RP/0/0/CPU0:Jan 28 22:01:36.356 : bgp[1056]: %ROUTING-BGP-5-ADJCHANGE : neighbor 10.10.10.4 Up (VRF:
 default) (AS: 4)
WARNING: Graceful Maintenance is Active
```

Use this warning as a reminder to deactivate graceful maintenance after the BGP convergence completes.