



BGP Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 24.1.x, 24.2.x, 24.3.x, 24.4.x

First Published: 2024-09-02

Last Modified: 2024-12-16

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

Preface xiii

Changes to This Document xiii

Communications, Services, and Additional Information xiii

CHAPTER 1

New and Changed BGP Features 1

New and Changed BGP Features 1

CHAPTER 2

YANG Data Models for BGP Features 3

Using YANG Data Models 3

CHAPTER 3

Implementing BGP 5

Prerequisites for Implementing BGP 7

BGP Functional Overview 8

BGP Router Identifier 8

BGP Route Distinguisher 9

Limitations for BGP route distinguisher 11

Configure static RD under VRF 11

Configure static RD under EVPN 12

Configure automatic RD under VRF 12

BGP Maximum Prefix - Discard Extra Paths 13

Configure Discard Extra Paths 13

Restrictions 16

BGP Labeled Unicast 17

BGP Labeled Unicast over RSVP-TE 21

Restrictions 22

Configure BGP-LU over RSVP-TE 23

Exclusion of Label Allocation for Non-Advertised Routes	28
How to exclude label allocation for non-advertised routes	29
EIBGP Policy-Based Multipath with Equal Cost Multipath	31
Restrictions for EIBGP Policy-Based Multipath with Equal Cost Multipath	33
Configure EIBGP Policy-Based Multipath with Equal Cost Multipath	34
ECMP Out of Resource Avoidance	36
Limitations for ECMP OOR Avoidance	38
Configure BGP for ECMP OOR Avoidance	38
Configure Dampening and DLB Modes	40
Protection of Directly Connected EBGp Neighbors through Interface-Based LPTS Identifier	46
Configure Protection of Directly Connected EBGp Neighbors through Interface-Based LPTS Identifier	47
Convergence for BGP Labeled Unicast PIC Edge	48
Black Box Monitoring	52
BGP Labeled Unicast Version 6	55
BGP Default Limits	58
BGP Next Hop Tracking	59
Next Hop as the IPv6 Address of Peering Interface	61
Scoped IPv4 Table Walk	62
Reordered Address Family Processing	62
New Thread for Next-Hop Processing	62
show, clear, and debug Commands	62
BGP Configuration	62
Neighbor Submode	64
Configuration Templates	64
Template Inheritance Rules	66
Viewing Inherited Configurations	70
No Default Address Family	76
Neighbor Address Family Combinations	76
Routing Policy Enforcement	76
Table Policy	77
BGP Update Group	77
BGP Update Generation and Update Groups	77
BGP Cost Community	77

How BGP Cost Community Influences the Best Path Selection Process	77
Cost Community Support for Aggregate Routes and Multipaths	78
Influencing Route Preference in a Multiexit IGP Network	79
Adding Routes to the Routing Information Base	80
BGP DMZ Aggregate Bandwidth	80
Configuring BGP DMZ Aggregate Bandwidth: Example	83
Configuring Policy-based Link Bandwidth: Example	83
64-ECMP Support for BGP	84
BGP Best Path Algorithm	84
Comparing Pairs of Paths	84
Order of Comparisons	86
Best Path Change Suppression	87
Administrative Distance	87
Route Dampening	89
Minimize Flapping	89
BGP Routing Domain Confederation	89
BGP Optimal Route Reflector	90
Use Case	90
VPN Route Limit on Route Reflectors	94
Guidelines and Recommendations	95
Limitations for VPN Route Limit	95
Configure VPN Route Limit	95
BGP Functional Overview	97
RPL - if prefix is-best-path/is-best-multipath	97
Remotely Triggered Blackhole Filtering with RPL Next-hop Discard Configuration	98
Configure Destination-based RTBH Filtering	98
Default Address Family for show Commands	100
TCP Maximum Segment Size	100
Per Neighbor TCP MSS	101
BGP Keychains	101
BGP Nonstop Routing	101
BGP Best-External Path	103
BGP Prefix Independent Convergence	103
Configure BGP PIC in Provider Edge Networks	105

Configure BGP PIC between Autonomous Systems	107
Command Line Interface (CLI) Consistency for BGP Commands	109
BGP Additional Paths	109
iBGP Multipath Load Sharing	110
Configure iBGP Multipath Load Sharing	110
Accumulated IGP Attribute for BGP	112
Accumulated Interior Gateway Protocol Attribute	114
BGP Accept Own	114
Configuring BGP Accept Own	116
BGP Link-State	117
Configuring BGP Link-state	119
Configuring Domain Distinguisher	120
BGP Permanent Network	121
Configuring BGP Permanent Network	122
Advertise Permanent Network	123
Selective FIB Download	125
Configure Selective FIB Download	126
BGP-RIB Feedback Mechanism for Update Generation	128
Delay BGP Route Advertisements	128
Configuration	129
Default-originate Under VRF	130
User-Defined Martian Address Check	130
BGP Multipath Enhancements	131
Overview of BGP Monitoring Protocol	132
Adj-RIB-In Post-Policy View for L3VPN Address Families	134
Local-RIB view for IP and L3VPN Address Families	137
Simultaneous Monitoring of Adj-RIB-In Pre-Policy and Post-Policy Views	139
Configure simultaneous monitoring of Adj-RIB-In Pre-Policy and Post-Policy views	140
BGP—Multiple Cluster IDs	145
BGP Flowspec Overview	146
Flow Specifications	147
Supported Matching Criteria and Actions	147
Limitations for BGP FlowSpec	154
BGP flowspec redirect from global VRF to L3VPN and segment routing policy	154

Traffic Filtering Actions	158
BGP Flowspec Client-Server Controller Model	159
Configure BGP Flowspec	160
Enabling BGP Flowspec for IPv6 Packet Length	173
BGP Extended Route Retention	176
How a CLUSTER_LIST Attribute is Used	177
Configure a Cluster ID per Neighbor	177
Disable Client-to-Client Reflection for Specified Cluster IDs	179
How to Implement BGP	180
Information About Implementing BGP	180
Adjust BGP Timers	180
Enabling BGP Routing	181
Configure Multiple BGP Instances for a Specific Autonomous System	184
Configure Routing Domain Confederation for BGP	185
Resetting an eBGP Session Immediately Upon Link Failure	188
BGP Fast Fallover	189
Configure BGP Fast-Fallover	190
Logging Neighbor Changes	191
Change BGP Default Local Preference Value	191
Configure MED Metric for BGP	192
Configure BGP Weights	193
Tune BGP Best-Path Calculation	195
Configure Aggregate Addresses	196
Indicate BGP Back-door Routes	198
Set BGP Administrative Distance	199
Configure BGP Neighbor Group and Neighbors	201
Configure Route Reflector for BGP	204
Understanding BGP MD5 Authentication	206
Redistributing iBGP Routes into IGP	206
Set BGP Administrative Distance	207
Configuring Discard Extra Paths	208
Configuring Per Neighbor TCP MSS	209
Disabling Per Neighbor TCP MSS	211
Configuring Discard Extra Paths	213

Configuring Per Neighbor TCP MSS	214
Disabling Per Neighbor TCP MSS	217
Configure BGP Route Filtering by Route Policy	219
Configure BGP Attribute Filtering	220
Configure BGP Next-Hop Trigger Delay	221
Disable Next-Hop Processing on BGP Updates	223
Configure BGP Community and Extended-Community Advertisements	224
Advertising IPv4 NLRI with IPv6 Next Hops in MP-BGP Networks	227
Configure Advertising IPv4 NLRI with IPv6 Next Hops in the Non-Default VRF	234
Configure BGP Cost Community	245
Configure Software to Store Updates from Neighbor	246
BGP Persistence	248
BGP Persistence Configuration: Example	249
Flexible BGP Persistence	250
Configure Flexible BGP Persistence	252
Configure LLGR Advertisement and Activation with Default and Peer-Specified Stale Time	252
Enable the LLGR Capability and Advertise It Only to iBGP Peers	253
BGP Graceful Maintenance	254
Restrictions for BGP Graceful Maintenance	254
Graceful Maintenance Operation	255
Inter Autonomous System	255
When to Shut Down After Graceful Maintenance	256
Activate Graceful Maintenance under BGP Router (All Neighbors)	256
Activate Graceful Maintenance on a Single Neighbor	258
Activate Graceful Maintenance on a Group of Neighbors	259
Direct Router to Reduce Route Preference	260
Bring Router or Link Back into Service	262
Show Command Outputs to Verify BGP Graceful Maintenance	262
Bring Router or Link Back into Service	263
Show Command Outputs to Verify BGP Graceful Maintenance	263
Flow-tag propagation	264
Restrictions for Flow-Tag Propagation	265
Source and destination-based flow tag	265
Configure Source and Destination-based Flow Tag	265

Configure Keychains for BGP	267
Configuring an MDT Address Family Session in BGP	268
Disable BGP Neighbor	272
Neighbor Capability Suppression	273
Configuration	273
BGP Dynamic Neighbors	274
Configuring BGP Dynamic Neighbors using Address Range	275
Remote AS List	276
Maximum-peers and Idle-watch timeout	277
Resetting Neighbors Using BGP Inbound Soft Reset	278
Resetting Neighbors Using BGP Outbound Soft Reset	279
Reset Neighbors Using BGP Hard Reset	279
Clearing Caches, Tables, and Databases	280
Display System and Network Statistics	281
Display BGP Process Information	283
Configure iBGP Multipath Load Sharing	284
Originate Prefixes with AiGP	285
Configure BGP Accept Own	287
Configuring BGP Link-state	290
Configuring BGP Permanent Network	291
How to Advertise Permanent Network	293
Enable BGP Unequal Cost Recursive Load Balancing	295
Configuring BGP Large Communities	298
Enabling BGP: Example	303
Displaying BGP Update Groups: Example	304
BGP Neighbor Configuration: Example	304
BGP Confederation: Example	305
BGP Route Reflector: Example	307
BGP Route Reflector: Example	307
BGP MDT Address Family Configuration: Example	308
BGP Nonstop Routing Configuration: Example	308
Best-External Path Advertisement Configuration: Example	308
Primary Backup Path Installation: Example	309
iBGP Multipath Loadsharing Configuration: Example	309

Discard Extra Paths Configuration: Example	309
Verify Per Neighbor TCP MSS: Examples	309
Originating Prefixes With AiGP: Example	312
BGP Accept Own Configuration: Example	312
BGP Unequal Cost Recursive Load Balancing: Example	313
Flow-tag propagation	315
Restrictions for Flow-Tag Propagation	315
Configuring Destination-Based Flow-Tag Propagation	316
Configure Software to Store Updates from Neighbor	319
Configuring BGP Route Dampening	320
Apply Policy When Updating Routing Table	322
Configure BGP Route Filtering by Route Policy	324
Configure Destination-based RTBH Filtering	325
Resilient Hashing and Flow Auto-Recovery	327
Persistent Loadbalancing	329
BGP Selective Multipath	330
Remove and Replace Private AS Numbers from AS Path in BGP	332
BGP DMZ Link Bandwidth for Unequal Cost Recursive Load Balancing	333
BGP Multi-Instance and Multi-AS	333
BGP Prefix Origin Validation Based on RPKI	334
Configure RPKI Cache-server	335
BGP Update Message Error Handling	337
BGP Attribute Filtering	337
BGP Error Handling and Attribute Filtering Syslog Messages	338
BGP-RIB Feedback Mechanism for Update Generation	338
Configure BGP Large Communities	339
Resetting an eBGP Session Immediately Upon Link Failure	344
User-Defined Martian Address Check	344
BGP Labeled Unicast MPLS IP POP Support	345
BGP eBGP Security GTSM	347
Steering of BGP Control-Plane Traffic over IP Path	348
Configure Steering of BGP Control Plane Traffic through an IP-only Path	350
128 Multipath ECMP	354
256 Multipath ECMP	358

Restrictions	358
Configure 256-Multipath BGP ECMP	358
Configure 1024 Equal Cost Multi-Path or 256 Unequal Cost Multi-Path using Hierarchical Load Balancing	361
Enabling Hierarchical Load Balancing	362
Verifying CEF Configuration	364
BGP Slow Peer	366
Restrictions	367
Types of Slow Peer	367
Use of Refresh Sub-Group for Dynamic Slow Peer Processing	367
Slow Peer State	368
Dynamic Slow Peer Identification Logic	368
Dynamic Slow Peer Recovery Logic	369
Refresh Sub-Group State	369
Slow Peer Configurations	370
Slow Peer Effective Configuration State	372
BGP Slow Peer Automatic Isolation from Update Group	374
Configuring an Aggregate Contributor	378
Setting BGP Attributes to the Aggregate Contributor	381
Management Information Base (MIBs) for BGP	383
Peering Between BGP Routers Within a Confederation	384
Restrictions for Peering Between BGP Routers Within the Same Confederation	386
Configure Peering Between BGP Routers Within the Same Confederation	386
Per-VRF label allocation for VPN routes	387
Limitations for Per-VRF label allocation for VPN routes	388
Configure Per-VRF label allocation for VPN routes	388
Configure Per-VRF label allocation for VPN routes - same RD scenario	388
Configure Per-VRF label allocation for VPN routes - different RD scenario	390
Virtual Routing Forwarding Next Hop Routing Policy	393
Configure VRF Next Hop Policy	393
Enhanced Monitoring of NSR Statistics	395
Enhanced BGP Monitoring and Prefix Analysis	399
Enhanced Monitoring of BGP Keepalive Messages	403
Enhanced Monitoring of BGP Memory Utilization	417

Enhanced Next Hop Monitoring	421
Enhanced Monitoring of Version-Rate Statistics	427
Flexible BGP Persistence	433



Preface

This preface contains these sections:

- [Changes to This Document, on page xiii](#)
- [Communications, Services, and Additional Information, on page xiii](#)

Changes to This Document

Table 1: Changes to This Document

Date	Change Summary
December 2024	Republished for Release 24.4.1.
September 2024	Republished for Release 24.3.1.
June 2024	Republished for Release 24.2.1.
February 2024	Initial release of this document

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

New and Changed BGP Features

- [New and Changed BGP Features, on page 1](#)

New and Changed BGP Features

This table lists the technical changes made to this document since it was first released.

Table 2: BGP Features Added or Modified in IOS XR Release 24.x.x

Feature	Description	Changed in Release	Where Documented
256-way ECMP	This feature was introduced.	Release 24.4.1	256 Multipath ECMP, on page 358
Per-VRF label allocation for VPN routes	This feature was introduced.	Release 24.4.1	Per-VRF label allocation for VPN routes, on page 387
Selective FIB download	This feature was introduced.	Release 24.4.1	Selective FIB Download, on page 125
Simultaneous monitoring of Adj-RIB-In Pre-Policy and Post-Policy views	This feature was introduced.	Release 24.4.1	Simultaneous Monitoring of Adj-RIB-In Pre-Policy and Post-Policy Views , on page 139
Flexible BGP Persistence	This feature was introduced.	Release 24.3.1	Flexible BGP Persistence
Advertising IPv4 NLRI with IPv6 Next Hops in MP-BGP Networks	This feature was modified.	Release 24.2.11	Advertising IPv4 NLRI with IPv6 Next Hops in MP-BGP Networks, on page 227
BGP Fast Fallover	This feature was introduced.	Release 24.2.11	BGP Fast Fallover, on page 189
ECMP Out of Resource Avoidance	This feature was introduced.	Release 24.2.11	ECMP Out of Resource Avoidance, on page 36

Feature	Description	Changed in Release	Where Documented
Steering of BGP Control-Plane Traffic over IP Path	This feature was introduced.	Release 24.2.11	Steering of BGP Control-Plane Traffic over IP Path, on page 348



CHAPTER 2

YANG Data Models for BGP Features

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

- [Using YANG Data Models, on page 3](#)

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 3

Implementing BGP

Border Gateway Protocol (BGP) is an Exterior Gateway Protocol (EGP) that allows you to create loop-free interdomain routing between autonomous systems. An *autonomous system* is a set of routers under a single technical administration. Routers in an autonomous system can use multiple Interior Gateway Protocols (IGPs) to exchange routing information inside the autonomous system and an EGP to route packets outside the autonomous system.

This module provides conceptual and configuration information on BGP.



Tip You can programmatically configure BGP and retrieve operational data using `openconfig-network-instance.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

- [Prerequisites for Implementing BGP, on page 7](#)
- [BGP Functional Overview, on page 8](#)
- [BGP Functional Overview, on page 97](#)
- [Disabling Per Neighbor TCP MSS, on page 211](#)
- [Configuring Discard Extra Paths, on page 213](#)
- [Configuring Per Neighbor TCP MSS, on page 214](#)
- [Disabling Per Neighbor TCP MSS, on page 217](#)
- [Configure BGP Route Filtering by Route Policy, on page 219](#)
- [Configure BGP Attribute Filtering, on page 220](#)
- [Configure BGP Next-Hop Trigger Delay, on page 221](#)
- [Disable Next-Hop Processing on BGP Updates, on page 223](#)
- [Configure BGP Community and Extended-Community Advertisements, on page 224](#)
- [Advertising IPv4 NLRI with IPv6 Next Hops in MP-BGP Networks, on page 227](#)
- [Configure BGP Cost Community, on page 245](#)
- [Configure Software to Store Updates from Neighbor, on page 246](#)
- [BGP Persistence, on page 248](#)
- [Flexible BGP Persistence, on page 250](#)
- [BGP Graceful Maintenance, on page 254](#)
- [Bring Router or Link Back into Service, on page 262](#)
- [Show Command Outputs to Verify BGP Graceful Maintenance, on page 262](#)
- [Bring Router or Link Back into Service, on page 263](#)
- [Show Command Outputs to Verify BGP Graceful Maintenance, on page 263](#)

- Flow-tag propagation, on page 264
- Neighbor Capability Suppression, on page 273
- BGP Dynamic Neighbors, on page 274
- Remote AS List, on page 276
- Maximum-peers and Idle-watch timeout, on page 277
- Resetting Neighbors Using BGP Inbound Soft Reset, on page 278
- Resetting Neighbors Using BGP Outbound Soft Reset, on page 279
- Reset Neighbors Using BGP Hard Reset, on page 279
- Clearing Caches, Tables, and Databases, on page 280
- Display System and Network Statistics, on page 281
- Display BGP Process Information, on page 283
- Configure iBGP Multipath Load Sharing , on page 284
- Originate Prefixes with AiGP, on page 285
- Configure BGP Accept Own, on page 287
- Configuring BGP Link-state, on page 290
- Configuring BGP Permanent Network, on page 291
- How to Advertise Permanent Network, on page 293
- Enable BGP Unequal Cost Recursive Load Balancing, on page 295
- Configuring BGP Large Communities, on page 298
- Enabling BGP: Example, on page 303
- Displaying BGP Update Groups: Example, on page 304
- BGP Neighbor Configuration: Example , on page 304
- BGP Confederation: Example , on page 305
- BGP Route Reflector: Example, on page 307
- BGP Route Reflector: Example, on page 307
- BGP MDT Address Family Configuration: Example, on page 308
- BGP Nonstop Routing Configuration: Example, on page 308
- Best-External Path Advertisement Configuration: Example, on page 308
- Primary Backup Path Installation: Example, on page 309
- iBGP Multipath Loadsharing Configuration: Example, on page 309
- Discard Extra Paths Configuration: Example, on page 309
- Verify Per Neighbor TCP MSS: Examples, on page 309
- Originating Prefixes With AiGP: Example, on page 312
- BGP Accept Own Configuration: Example, on page 312
- BGP Unequal Cost Recursive Load Balancing: Example, on page 313
- Flow-tag propagation, on page 315
- Restrictions for Flow-Tag Propagation, on page 315
- Configuring Destination-Based Flow-Tag Propagation, on page 316
- Configure Software to Store Updates from Neighbor, on page 319
- Configuring BGP Route Dampening, on page 320
- Apply Policy When Updating Routing Table, on page 322
- Configure BGP Route Filtering by Route Policy, on page 324
- Configure Destination-based RTBH Filtering , on page 325
- Resilient Hashing and Flow Auto-Recovery, on page 327
- Persistent Loadbalancing , on page 329
- BGP Selective Multipath, on page 330

- [Remove and Replace Private AS Numbers from AS Path in BGP](#), on page 332
- [BGP DMZ Link Bandwidth for Unequal Cost Recursive Load Balancing](#), on page 333
- [BGP Multi-Instance and Multi-AS](#), on page 333
- [BGP Prefix Origin Validation Based on RPKI](#), on page 334
- [BGP Update Message Error Handling](#), on page 337
- [BGP Attribute Filtering](#), on page 337
- [BGP Error Handling and Attribute Filtering Syslog Messages](#), on page 338
- [BGP-RIB Feedback Mechanism for Update Generation](#), on page 338
- [Configure BGP Large Communities](#), on page 339
- [Resetting an eBGP Session Immediately Upon Link Failure](#), on page 344
- [User-Defined Martian Address Check](#), on page 344
- [BGP Labeled Unicast MPLS IP POP Support](#), on page 345
- [BGP eBGP Security GTSM](#), on page 347
- [Steering of BGP Control-Plane Traffic over IP Path](#), on page 348
- [128 Multipath ECMP](#), on page 354
- [256 Multipath ECMP](#), on page 358
- [Configure 1024 Equal Cost Multi-Path or 256 Unequal Cost Multi-Path using Hierarchical Load Balancing](#), on page 361
- [BGP Slow Peer](#), on page 366
- [BGP Slow Peer Automatic Isolation from Update Group](#), on page 374
- [Configuring an Aggregate Contributor](#), on page 378
- [Management Information Base \(MIBs\) for BGP](#), on page 383
- [Peering Between BGP Routers Within a Confederation](#), on page 384
- [Per-VRF label allocation for VPN routes](#), on page 387
- [Virtual Routing Forwarding Next Hop Routing Policy](#), on page 393
- [Enhanced Monitoring of NSR Statistics](#), on page 395
- [Enhanced BGP Monitoring and Prefix Analysis](#), on page 399
- [Enhanced Monitoring of BGP Keepalive Messages](#), on page 403
- [Enhanced Monitoring of BGP Memory Utilization](#), on page 417
- [Enhanced Next Hop Monitoring](#), on page 421
- [Enhanced Monitoring of Version-Rate Statistics](#), on page 427
- [Flexible BGP Persistence](#), on page 433

Prerequisites for Implementing BGP

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- The current Internet BGP table contains approximately 1.1 million IPv4 routes and 200,000 IPv6 routes. With an average of two paths per route, the BGP process typically requires around 5.5 GB of RAM to manage the full Internet BGP table. As the IPv6 Internet table continues to expand, the memory requirements for the BGP process are expected to increase. Therefore, Cisco recommends using the Service Edge (SE) version of Route Processor (RP) or Route Switch Processor (RSP) cards, or fixed chassis, on routers that will maintain a full BGP table

BGP Functional Overview

Table 3: Feature History Table

Feature Name	Release Information	
BGP Functional Overview	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700) (select variants only*)</p> <p>Enhances routing efficiency by enabling Border Gateway Protocol (BGP) to exchange routing information between autonomous systems. Supports route filtering, route aggregation, and policy-based routing, allowing for precise control over routing decisions. The feature is essential for large-scale network environments requiring scalability and stability. It includes improvements in route reflection and confederation to optimize routing paths and decrease the burden on network resources. Additionally, it facilitates seamless integration with existing network infrastructure, ensuring reliable and efficient data transfer across diverse networks.</p> <p>*This functionality is now supported on Cisco 8712-MOD-M routers.</p>

BGP uses TCP as its transport protocol. Two BGP routers form a TCP connection between one another (peer routers) and exchange messages to open and confirm the connection parameters.

BGP routers exchange network reachability information. This information is mainly an indication of the full paths (BGP autonomous system numbers) that a route should take to reach the destination network. This information helps construct a graph that shows which autonomous systems are loop free and where routing policies can be applied to enforce restrictions on routing behavior.

Any two routers forming a TCP connection to exchange BGP routing information are called peers or neighbors. BGP peers initially exchange their full BGP routing tables. After this exchange, incremental updates are sent as the routing table changes. BGP keeps a version number of the BGP table, which is the same for all of its BGP peers. The version number changes whenever BGP updates the table due to routing information changes. Keepalive packets are sent to ensure that the connection is alive between the BGP peers and notification packets are sent in response to error or special conditions.



Note ASN change for BGP process is not currently supported via **commit replace**.

BGP Router Identifier

For BGP sessions between neighbors to be established, BGP must be assigned a router ID. The router ID is sent to BGP peers in the OPEN message when a BGP session is established.

BGP attempts to obtain a router ID in the following ways (in order of preference):

- By means of the address configured using the **bgp router-id** command in router configuration mode.
- By using the highest IPv4 address on a loopback interface in the system if the router is booted with saved loopback address configuration.

- By using the primary IPv4 address of the first loopback address that gets configured if there are not any in the saved configuration.

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, an error message is entered in the system log, and the **show bgp summary** command displays a router ID of 0.0.0.0.

After BGP has obtained a router ID, it continues to use it even if a better router ID becomes available. This usage avoids unnecessary flapping for all BGP sessions. However, if the router ID currently in use becomes invalid (because the interface goes down or its configuration is changed), BGP selects a new router ID (using the rules described) and all established peering sessions are reset.



Note We strongly recommend that the **bgp router-id** command is configured to prevent unnecessary changes to the router ID (and consequent flapping of BGP sessions).

BGP Route Distinguisher

A BGP Route Distinguisher (RD) is a network identifier that

- uniquely distinguishes VPN routes to prevent conflicts,
- enables separate route tables for different VPNs, and
- supports scalable, secure, and efficient VPN route management.

Table 4: Feature History Table

Feature Name	Release	Description
BGP Route Distinguisher	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100]) This feature is now extended to Cisco 8712-MOD-M routers.
BGP Route Distinguisher	Release 7.0.1	You can now enhance route differentiation by uniquely identifying VPN routes, facilitating seamless integration across different networks. This feature assigns unique identifiers to VPN routes, ensuring no route conflicts and enabling efficient route filtering and management.

In network design solutions where customer equipment is dual-homed and Fast Reroute is required, such as in EVPN and BGP Prefix Independent Convergence (PIC) Edge solutions, the RD associated with each VRF must be unique per Provider Edge (PE) router. In other design scenarios, while it isn't mandatory for the RD to be unique per PE, it is highly recommended to make it unique. This practice facilitates easier transitions to dual-homed solutions in the future.

Ensuring unique RDs significantly enhances network scalability and security by maintaining distinct route tables for separate VPNs, which prevents routing conflicts and optimizes network performance. Additionally, it simplifies routing policies, making the configuration and management of large-scale networks more efficient and reliable.

To maintain unique RDs across the network, you can configure them either as automatic or static RDs under VRF or EVPN services such as EVI, VNI, or SRv6.

You can configure RD in these formats:

- **IP:nn**
 - **IP** is the BGP Router-ID configured on the router.
 - **nn** is the EVI, VNI, or SRv6, or a value selected from a pool if the default value is already in use.
- **2B:4B**
 - **2B** is the two byte administrator field.
 - **4B** is the four byte assigned number field.
- **4B:2B**
 - **4B** is the four byte administrator field.
 - **2B** is the two byte assigned number field.

The following table provides a comparison between static and automatic RD configurations in VRF and EVPN technologies.

Table 5: Comparison between static and automatic RD under VRF and EVPN

Technologies	Automatic RD	Static RD
VRF	You can configure automatic RD under VRF using the rd auto command. For more information, see Configure automatic RD under VRF, on page 12 .	You can configure static RD under VRF in the IP:nn, 2B:4B, and 4B:2B format. For more information, see Configure static RD under VRF, on page 11 .
EVPN	Cisco IOS XR software automatically assigns a unique RD to the EVPN EVI in the IP:nn format. For example, 192.0.2.1:100.	You can configure static RD under EVPN in the IP:nn, 2B:4B, and 4B:2B format. It is recommended to configure static RD under EVPN in the 2B:4B or 4B:2B format, as the IP:nn format may conflict with EVPN automatic RD. For more information, see Configure static RD under EVPN, on page 12 .

Limitations for BGP route distinguisher

These are the limitation for BGP RD:

- When you configure static RD in IP:nn format, make sure that it doesn't conflict with the IP:nn RD automatically generated value of another EVPN service such as EVI, VNI, or SRV6 in your network.
If the configured static RD value matches the autogenerated RD in IP:nn format for this or another service, then:
 - The Cisco IOS XR software rejects the static RD configuration in the following scenario:
 - If the EVI is configured first under a bridge-domain or xconnect and
 - then, a matching static RD is configured under the EVI.
 - Cisco IOS XR software displays a system log message and doesn't reject the static RD in the following scenario:
 - The RD is configured first under the EVI and
 - then, the EVI is associated with a bridge-domain or xconnect.
- In a Data Center Interconnect (DCI) deployment, you must not use the same RD between the edge DCI gateway and the MPLS-VPN PE router, nor between the edge DCI gateway and the Leaf with Top of Rack (ToR) switch when you re-originate routes with a stitching route target (stitching-rt) for a specific Virtual Routing and Forwarding (VRF) instance.

Configure static RD under VRF

Perform these steps to configure static RD under VRF:

Procedure

Step 1 Configure a VRF to isolate routing tables and segment traffic.

Example:

```
Router# configure
Router(config)# vrf VRF1
```

Step 2 Configure a static RD under VRF to differentiate and isolate routes in one of the following formats:

• **IP:nn**

```
Router(config-vrf)# rd 192.0.2.1:100
Router(config-vrf)# commit
```

• **2B:4B**

```
Router(config-vrf)# rd 1:100
Router(config-vrf)# commit
```

• **4B:2B**

```
Router(config-vrf)# rd 65536:1
Router(config-vrf)# commit
```

Configure static RD under EVPN

When you configure static RD under EVPN, use the 2B:4B or 4B:2B format, as the IP:nn format may conflict with EVPN automatic RD. For more information, see [Limitations for BGP route distinguisher, on page 11](#).

Perform these steps in the given order to configure static RD under EVPN and to prevent the conflict between static RD and automatic RD values:

Procedure

Step 1 Configure an EVPN EVI to isolate L2 or L3 networks.

Example:

```
Router# configure
Router(config)# evpn evi 100
```

Step 2 Configure a static RD under EVPN EVI to differentiate and isolate routes in one of the following formats:

• **2B:4B**

```
Router(config-evpn)# rd 1:100
Router(config-evpn)# commit
```

• **4B:2B**

```
Router(config-evpn)# rd 65536:1
Router(config-evpn)# commit
```

Configure automatic RD under VRF

Perform these steps to configure automatic RD under VRF.

Procedure

Step 1 Configure a VRF to isolate routing tables and segment traffic.

Example:

```
Router# configure
Router(config)# vrf VRF1
```

Step 2 Configure an automatic RD under the VRF to let the Cisco IOS XR software automatically assign a unique RD for differentiating and isolating routes.

Example:

```
Router(config-vrf)# rd auto
Router(config-vrf)# commit
```

BGP Maximum Prefix - Discard Extra Paths

IOS XR BGP maximum-prefix feature imposes a maximum limit on the number of prefixes that are received from a neighbor for a given address family. Whenever the number of prefixes received exceeds the maximum number configured, the BGP session is terminated, which is the default behavior, after sending a cease notification to the neighbor. The session is down until a manual clear is performed by the user. The session can be resumed by using the **clear bgp** command. It is possible to configure a period after which the session can be automatically brought up by using the **maximum-prefix** command with the **restart** keyword. The maximum prefix limit can be configured by the user.



Note Starting IOS-XR Release 7.3.1, the router does not apply default limits if the user does not configure the maximum number of prefixes for the address family.

Discard Extra Paths

An option to discard extra paths is added to the maximum-prefix configuration. Configuring the discard extra paths option drops all excess prefixes received from the neighbor when the prefixes exceed the configured maximum value. This drop does not, however, result in session flap.

The benefits of discard extra paths option are:

- Limits the memory footprint of BGP.
- Stops the flapping of the peer if the paths exceed the set limit.

When 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 is flapped.

On the same lines, the following describes the actions when the maximum prefix value is changed:

- If the maximum value alone is changed, a route-refresh message is sourced, if applicable.
- If the new maximum value is greater than the current prefix count state, the new prefix states are saved.
- If the new maximum value is less than the current prefix count state, then some existing prefixes are deleted to match the new configured state value.

There is currently no way to control which prefixes are deleted.

Configure Discard Extra Paths

The discard extra paths option in the maximum-prefix configuration allows you to drop all excess prefixes received from the neighbor when the prefixes exceed the configured maximum value. This drop does not, however, result in session flap.

The benefits of discard extra paths option are:

- Limits the memory footprint of BGP.

- Stops the flapping of the peer if the paths exceed the set limit.

When 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 is flapped.


Note

- When the router drops prefixes, it is inconsistent with the rest of the network, resulting in possible routing loops.
- If prefixes are dropped, the standby and active BGP sessions may drop different prefixes. Consequently, an NSR switchover results in inconsistent BGP tables.
- The discard extra paths configuration cannot co-exist with the *soft reconfig* configuration.
- When the system runs out of physical memory, bgp process exits and you must manually restart bpm. To manually restart, use the **process restart bpm** command.

Perform this task to configure BGP maximum-prefix discard extra paths.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **address-family** { **ipv4** | **ipv6** } **unicast**
5. **maximum-prefix** *maximum* **discard-extra-paths**
6. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

Step 1 **configure**
Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters XR Config mode.

Step 2 **router bgp** *as-number*
Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 10
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address*
Example:

```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **address-family { ipv4 | ipv6 } unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
```

Specifies either the IPv4 or IPv6 address family and enters address family configuration submenu.

Step 5 **maximum-prefix *maximum* discard-extra-paths**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# maximum-prefix 1000 discard-extra-paths
```

Configures a limit to the number of prefixes allowed.

Configures discard extra paths to discard extra paths when the maximum prefix limit is exceeded.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Example

The following example shows how to configure discard extra paths feature for the IPv4 address family:

```
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# router bgp 10
RP/0//CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0//CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0//CPU0:router(config-bgp-nbr-af)# maximum-prefix 1000 discard-extra-paths
RP/0//CPU0:router(config-bgp-vrf-af)# commit
```

The following screen output shows details about the discard extra paths option:

```
RP/0//CPU0:ios# show bgp neighbor 10.0.0.1

BGP neighbor is 10.0.0.1
Remote AS 10, local AS 10, internal link
Remote router ID 0.0.0.0
BGP state = Idle (No best local address found)
Last read 00:00:00, 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 00:00:00, attempted 0, written 0
Second last write 00:00:00, attempted 0, written 0
Last write before reset 00:00:00, attempted 0, written 0
```

Restrictions

- When the router drops prefixes, it is inconsistent with the rest of the network, resulting in possible routing loops.
- If prefixes are dropped, the standby and active BGP sessions may drop different prefixes. Consequently, an NSR switchover results in inconsistent BGP tables.
- The discard extra paths configuration cannot co-exist with the *soft reconfig* configuration.

BGP Labeled Unicast

Table 6: Feature History Table

Feature Name	Release	Description
BGP Labeled Unicast	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700)(select variants only*).</p> <p>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.</p> <p>*This feature is now supported on Cisco 8712-MOD-M Routers.</p>

The BGP Labeled Unicast (LU) feature, also known as unified MPLS, provides MPLS transport between Provider Edge (PE) routers that are separated by either many IGP boundaries (intra-AS) or by many autonomous systems (inter-AS). Using autonomous systems border routers (ASBRs), you can advertise loopback prefixes of PEs and their MPLS label bindings: iBGP between area border routers (ABRs) and eBGP between autonomous system border routers. You can use Multihop eBGP between the PEs if they are in different autonomous systems (ASes) to exchange the VPN routes. You can run 6PE and other services between the PEs that have BGP LU connectivity.

The BGP LU feature lowers the IGP labeled prefix scale and adjacency scale values. If the router is not being configured with BGP LU, it is necessary to prevent lowering of scale values. Hence it is mandatory to configure the hw-module command before you enable the BGP LU feature. Restart the router for the hw-module command configuration to take effect.

Restrictions

- Cisco 8000 supports only per-vrf label mode.
- You can use LDP or Segment Routing (SR) as the transport underlay. You cannot use TE as the transport underlay.
- BGP PIC edge feature is not supported.
- L3VPN and 6VPE over BGP LU feature is not supported.
- BGP PIC core feature is supported.
- The **label-allocation-mode** is deprecated from release 7.4.1. The function of this command can be carried out using label mode command under configured [address-family](#).

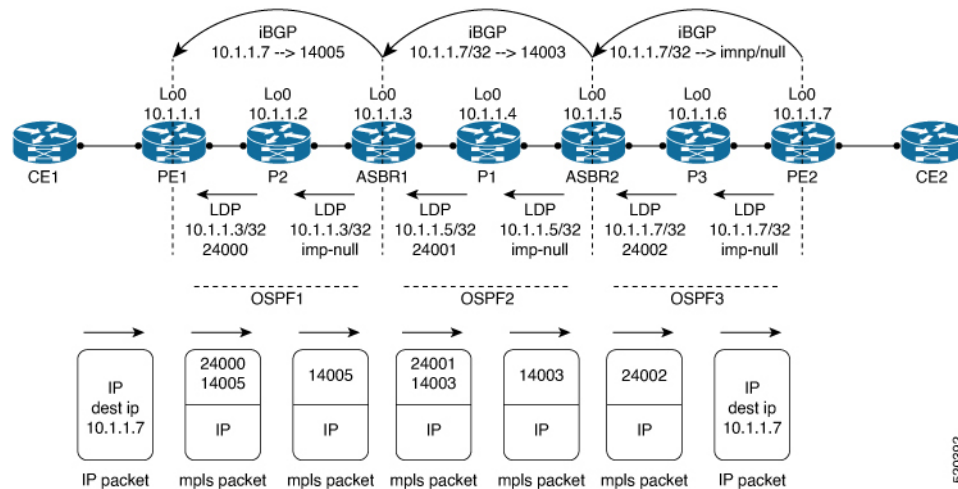
Supported features

The following features are supported:

- BGP LU with inter-AS option C
- 6PE over MPLS transport using LDP or Segment Routing.
- BGP PIC core

Topology

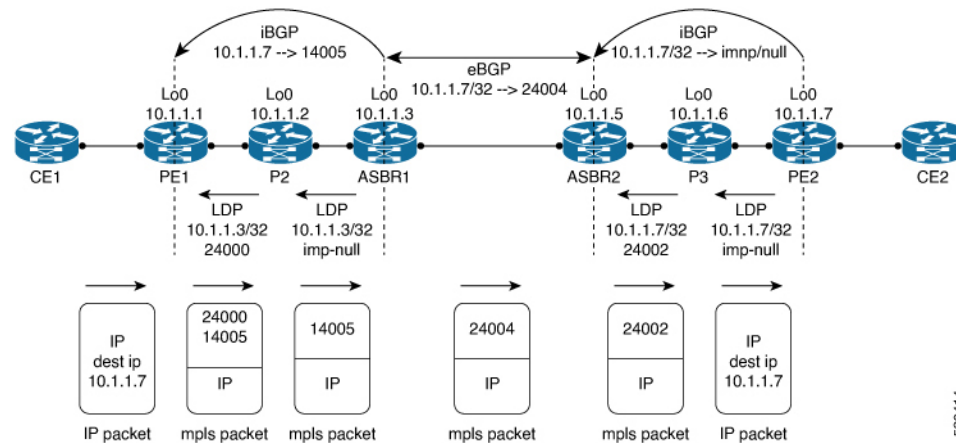
Figure 1: BGP Labeled Unicast (Intra-Autonomous System) Control Plane and Data Plane



The above diagram explains how PE1 is connected with PE2 through MPLS connectivity. PE1 and PE2 are separated by many areas within the same AS. Consider three network areas OSPF1, OSPF2, and OSPF3. Each of these areas is running separate OSPFs. LDP acts as transport between each of these areas. To establish a connection between the Provider Edge routers PE1 and PE2, send iBGP from PE2 to PE1 through P3, ASBR2, P1 and ASBR1, P2. PE1 must learn the loopback address of PE2 to establish a connection between the loopback address of PE1 and the loopback address of PE2.

The loopback address of PE2 which is 10.1.1.7 advertises a BGP label through iBGP to ASBR2. This address is advertised as an implicit null label. The ASBR2 allocates a local label 14003 for the loopback address 10.1.1.7 and sends it to ASBR1. ASBR1 allocates its own label 14005 to the loopback address 10.1.1.7 and sends it to PE1. PE1 has learnt the prefix of loopback address 10.1.1.7 and the BGP label 14005. The BGP next hop for PE1 is ASBR1. When PE1 sends traffic to PE2, PE1 adds two labels: the BGP-LU label and transport LDP label. The transport LDP label 24000, is above the BGP-LU label 14005. PE1 imposes the transport LDP label and the BGP-LU label when PE1 transmits an IP packet destined to the loopback address 10.1.1.7. The transport LDP label carries the packet to ASBR1. ASBR1 receives the IP packet. It contains only the BGP-LU label, 14005. ASBR1 swaps the BGP-LU label from 14005 to 14003 and imposes transport LDP label 24001 and sends the IP packet to ASBR2. ASBR2 receives the packet. The BGP-LU label for the loopback address 10.1.1.7 in ASBR2 is implicit null. Only the transport label is pushed to 24002. ASBR2 transmits the transport label that carries the transport to PE2.

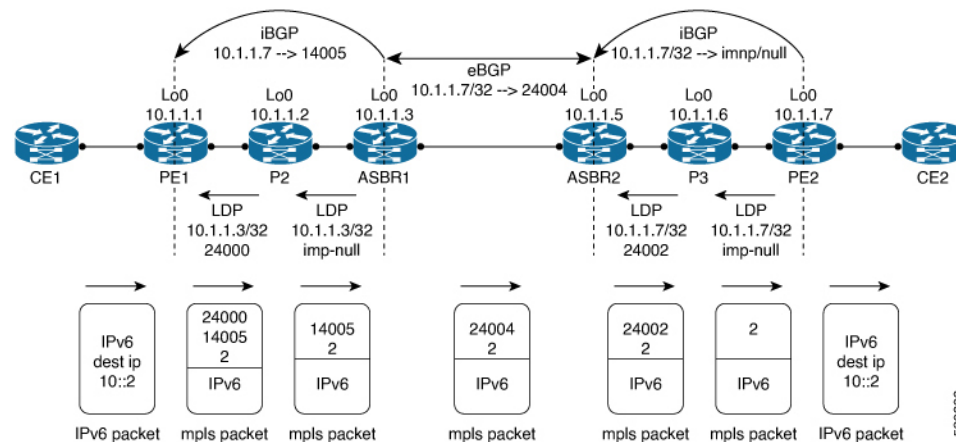
Figure 2: BGP Labeled Unicast (Intra-Autonomous System Option C) Control Plane and Data Plane



ASBR2 prefers IGP MPLS path over BGP path 10.1.1.7. It advertises LDP local label as BGP label to ASBR1. A LDP swap operation takes place on ASBR2.

The above figure explains how PE1 is connected with PE2 through MPLS connectivity using eBGP. In the above-mentioned scenario, eBGP exists between ASBR1 and ASBR2. PE2 advertises the BGP-LU label which has a value of implicit null to ASBR2 through iBGP. The loopback address is known to ASBR2 through the IGP. ASBR2 prefers the IGP path with ldp label 24002. ASBR2 allocates local label 24004 to loopback 10.1.1.7. It advertises the local label 24004 to ASBR1. ASBR1 creates a local label 14005 and advertises it to PE1. Now, PE1 is aware of the loopback address 10.1.1.7. The IP packet has two labels: the BGP label 14005 and the transport label 24000. PE1 transmits the IP packet to ASBR1. The IP packet received by ASBR1 has only the BGP LU label 14005. ASBR1 swaps BGP-LU label from 14005 to 24004. The IP packet reaches ASBR2 where LDP label 24002 is pushed and transmits the packet to PE2.

Figure 3: 6PE over BGP LU (Inter-AS Option C) Control Plane and Data Plane



The above illustration explains how PE1 is connected with PE2 through MPLS connectivity using Multihop eBGP between multiple ASes. Multihop BGP exists between PE1 and PE2. PE1 and PE2 can exchange 6PE routes on the multihop eBGP with the labels. The label value for 6PE is v6 explicit null. When PE2 advertises v6 prefix 10::2/128, the label is always the explicit null label. The BGP label and LDP label constitute the top two labels. The 6PE label constitutes the bottom label which is v6 explicit null. The v6 packet reaches PE1 with destination IP 10:2. The label imposition takes place here. The 6PE label of value 2 is imposed first, the BGP label 14005 is imposed next, and then the next hop LDP label 14005 for the BGP LU next hop is

imposed. ASBR1 swaps BGP-LU label from 14005 to 24004 and forwards the packet to ASBR2. ASBR2 adds LDP label on top of 6PE label 2 and forwards it to P3 where LDP label is POPed, so PE2 receives packet with 6PE explicit null label only. PE2 performs a v6 lookup and forwards the packet.

Configure BGP Labeled Unicast

```
Router(config)# hw-module profile cef bgplu enable
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
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

/* Note: Restart the router for the hw-module command configuration to take effect. */
```

Running Configuration

```
!
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
!
```

Verification

SME to provide the show output required below.

```
Router # show bgp ipv6 unicast labels
Network                Next Hop                Rcvd Label                Local Label

Router# show bgp ipv6 unicast labels
Network                Next Hop                Rcvd Label                Local Label
```

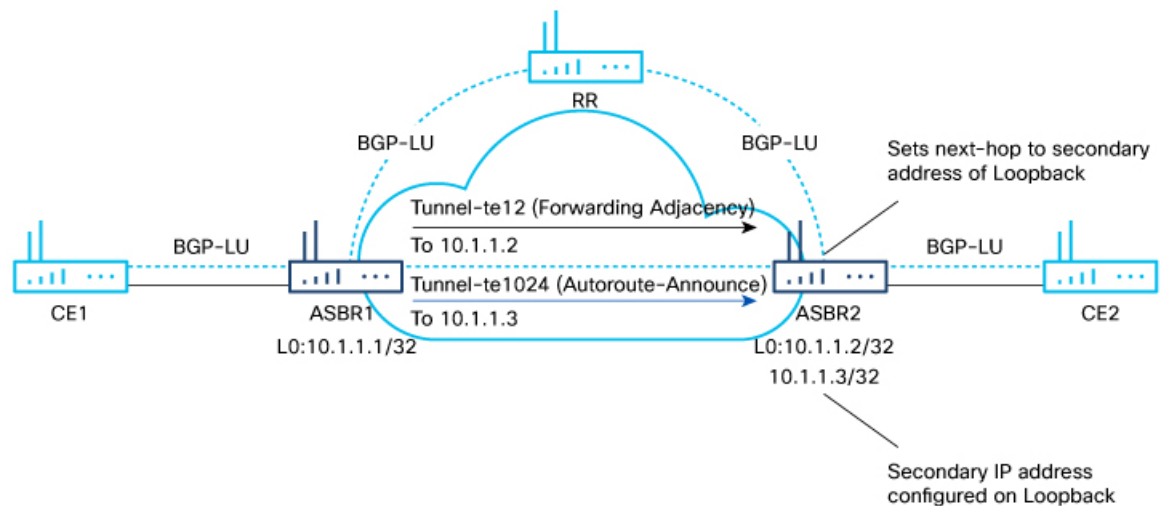
BGP Labeled Unicast over RSVP-TE

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
BGP Labeled Unicast over RSVP-TE	Release 24.4.1	<p>Introduced in this release on: Fixed Systems(8200, 8700[ASIC:K100]); Modular Systems (8800 [LC ASIC: P100]) (select variants only*).</p> <p>*This feature is now supported on:</p> <ul style="list-style-type: none"> • 8212-32FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 8712-MOD-M
BGP Labeled Unicast over RSVP-TE	Release 7.11.1	<p>You can now steer the MPLS traffic as per your requirement instead of relying on what the IGP directs.</p> <p>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.</p> <p>The feature introduces these changes:</p> <p>CLI:</p> <ul style="list-style-type: none"> • hw-module profile cef bgplu-over-rsvp-te enable <p>YANG Data Models:</p> <ul style="list-style-type: none"> • Cisco-IOS-XR-npu-hw-profile-cfg.yang (see GitHub, YANG Data Models Navigator)

BGP Labeled Unicast over RSVP-TE feature enables the routers to forward the BGP labeled unicast traffic to the BGP-LU next hop router through Reservation Protocol - Traffic Engineering (RSVP-TE) tunnels. With this feature, you can choose the tunnel (path) to transport the traffic as per your requirement. For example, Autoroute Announce (AA) tunnels can be used exclusively for the traffic that is sent to the tunnel destination address. All other traffic, by default, is routed through the Forwarding-Adjacency (FA) tunnels.

Figure 4: BGP Labeled Unicast over RSVP-TE



- All traffic destined for 10.1.1.3/32 will be steered into the AA tunnel (Tunnel-te1024)
- All other traffic to 10.1.1.2/32 will be steered into the FA tunnel (Tunnel-te12)

523494

In this example, BGP-LU connects ASBR1 with CE1, and ASBR2 with CE2. ASBR1 is configured with two RSVP-TE tunnels to ASBR2. The tunnel configured with FA is connected to the primary IP address (10.1.1.2), and the tunnel configured with AA is connected to the secondary IP address (10.1.1.3). ASBR2 sets the next-hop of the BGP-LU prefixes received from CE2 with the secondary IP address (10.1.1.3). So, ASBR1 uses the AA tunnel to forward packets that are destined for CE2.



Note If a BGP-LU route is learned via two BGP-NH routers, and if one BGP-NH router is reachable via RSVP-TE, and the other BGP-NH router is reachable via regular next-hop, the path reachable via regular next-hop is selected for forwarding.

Fast Reroute (FRR) mechanism provides protection to the transported traffic against link and node failures.

Restrictions

The following restrictions apply for the BGP-LU over RSVP-TE feature:

- Configuring BGP-LU over RSVP-TE along with BGP-LU (over NH) and Class-based forwarding (CBF) is not allowed. You must disable BGP-LU and CBF configurations before enabling the BGP-LU over RSVP-TE feature. Otherwise, the router displays an error message.
- BGP-LU over RSVP-TE feature is not supported on Q100-based line cards.
- BGP-LU over SR-TE is not supported.
- Services such as L3VPN, 6PE, and 6VPE are not supported.
- You can use LDP or Segment Routing (SR) as the transport underlay. But, you cannot use TE as the transport underlay.
- Reaching ASBR (BGP-NH) through regular NH and RSVP-TE is not supported.

Configure BGP-LU over RSVP-TE

Configuration Example

This example shows how to configure the BGP-LU over RSVP-TE feature.

```
/* Disable BGP-LU and CBF.*/
Router(config)# no hw-module profile cef bgplu enable
Router(config)# no hw-module profile cef cbf enable

/* Enable BGP-LU over RSVP-TE.*/
Router(config)# hw-module profile cef bgplu-over-rsvpte enable
```



Note By default, this feature supports a maximum of 1k tunnels. To increase the capacity to support 5k tunnels, run the **hw-module profile cef te-tunnel highscale-no-ldp-over-te** command.

```
/* Configure loopback interfaces. */
Router(config)# interface Loopback1001
Router(config-if)# ipv4 address 10.10.10.10 255.255.255.255
Router(config-if)# exit
Router(config)# interface tunnel-te1
Router(config-if)# ipv4 unnumbered Loopback0
Router(config-if)# autoroute announce
Router(config-if)# exit
Router(config-if)# destination 10.10.10.11
Router(config-if)# path-option 1 dynamic

/* Configure BGP.*/
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

/* Configure BGP Neighbor.*/
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

/* Configure MPLS LDP.*/
Router(config)# mpls ldp
Router(config-ldp)# router-id 10.1.1.1
Router(config-ldp)# interface tunnel-te1
Router(config-ldp)# exit
```



Note Reload the router for the **hw-module** commands to take effect.

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

```

Verification

Verify the details of route paths:

```

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

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.

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

Verify the details of LSP tunnel:

Router# **show mpls forwarding prefix 209.165.200.225/27**

Tue Jun 6 14:00:17.601 UTC

Local Label	Outgoing Label	Prefix or ID	Outgoing Interface	Next Hop	Bytes Switched
36112	34184	209.165.200.225/27		10.1.4.1	0
	39146	209.165.200.225/27		10.1.5.1	0

Verify the contents of the Fast Reroute (FRR) database:

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

Verify the forwarding information on tunnels:

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
```

Verify the utilization of banks in the NPU resources:

```
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
current usage      : 48152
```

Exclusion of Label Allocation for Non-Advertised Routes

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
Exclusion of Label Allocation for Non-Advertised Routes	Release 7.10.1	<p>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.</p> <p>Prior to this release, label allocation was done regardless of whether the routes being advertised. This resulted in inefficient use of label space.</p>

The functionality to control label allocation to the routes which are not advertised to peers is introduced. You can now choose to assign labels to the routes which are advertised to the peers.

Provider Edge (PE) routers works as autonomous systems border routers (ASBRs) where this feature is configured.

You can set the **community** attribute to either **no-advertise** or **no-export** in route-policy configuration mode to the routes which are not going to be advertised to peers. Once the **community** attribute in the route-policy is updated, the router doesn't allocate any label to those routes.



Note **no-export** is only for eBGP and **no-advertise** can be used for both eBGP and iBGP.

How to exclude label allocation for non-advertised routes

Configuration Example

This example shows how to set the *community* parameter to **no-advertise** for the routes which are not going to be advertised to any peer routes.

```
/*Configure the community set*/
Router(config)#community-set no-advertise
Router(config-comm)#no-advertise
Router(config-comm)#end-set

/*Configure the route policy*/
Router(config)#route-policy set-no-advertise
Router(config-rpl)#set community no-advertise additive
Router(config-rpl)#end-policy
Router(config-bgp-af)#route-policy pass_all
Router(config-rpl)# pass
Router(config-rpl)#end-policy
Router(config)#route-policy pass_all
Router(config-rpl)# pass
Router(config-rpl)#end-policy

/*Apply the route policy as inbound route policy*/
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
```

Running Configuration

```
community-set no-advertise
  no-advertise
end-set
!
!
route-policy set-no-advertise
  set community no-advertise additive
end-policy
!
!
route-policy pass_all
  pass
end-policy
!
```

Verification

Use **show bgp vpnv6 unicast rd** command to verify the **community** parameter is set to **no-advertised**.

```
Router(config)# 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
```

EIBGP Policy-Based Multipath with Equal Cost Multipath

Table 9: Feature History Table

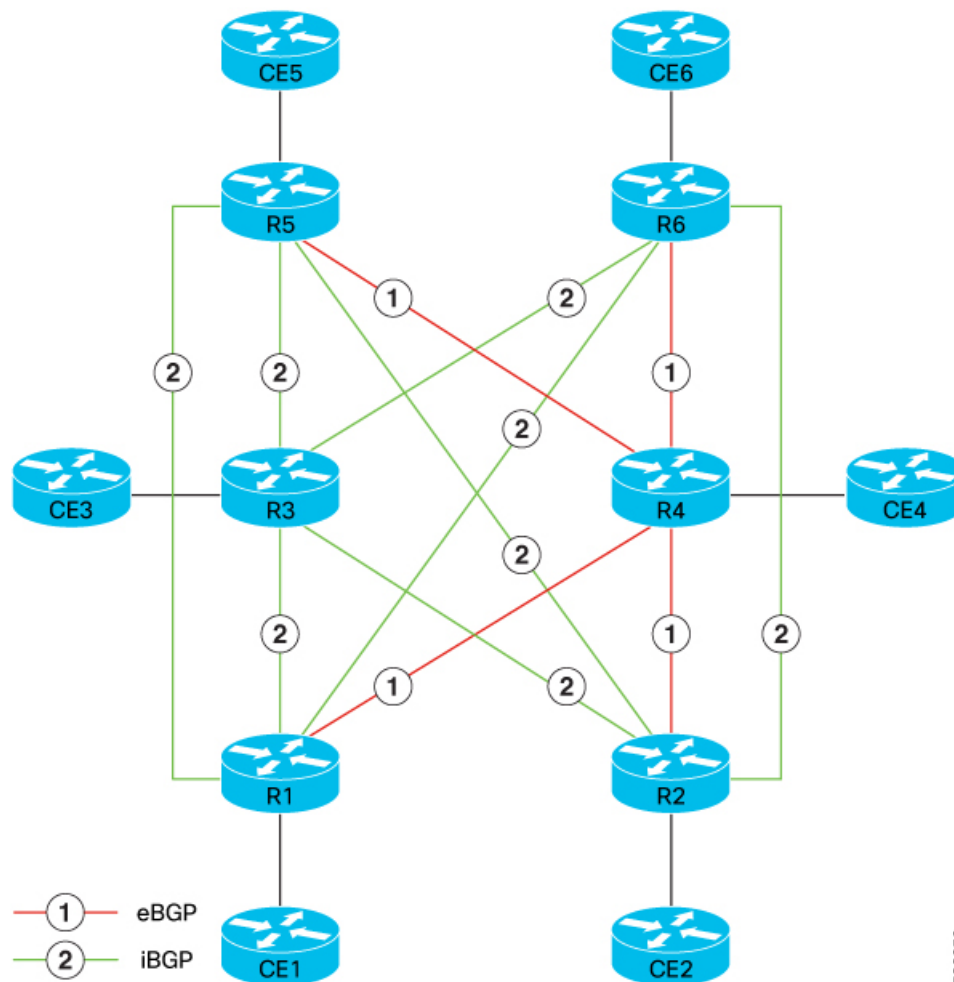
Feature Name	Release Name	Description
EIBGP Policy-Based Multipath with Equal Cost Multipath	Release 7.10.1	<p>You can gain control over traffic distribution and load-balancing capabilities by including policy-based multipath selection across various BGP variations, including iBGP, eBGP, and eiBGP. This is achieved through the utilization of BGP communities, nexthops, and path types.</p> <p>Additionally, by employing the equal cost multipath (ECMP) option in eiBGP, this feature provides the capability to select ECMP across the iBGP paths chosen for eiBGP.</p> <p>The feature introduces these changes:</p> <p>CLI:</p> <p>The keywords route-policy and equal-cost are added to the command:</p> <p>maximum-paths</p> <p>YANG Data Model:</p> <ul style="list-style-type: none"> Cisco-IOS-XR-um-router-bgp-cfg <p>(see GitHub, YANG Data Models Navigator)</p>

Overview

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 multipath (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.

Topology



This topology illustrates a network comprising BGP peers denoted as R1 through R6. Consider a scenario, there is specific need wherein you are in the process of transitioning from utilizing 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.

Expected Behavior

In the context of CE routers (CE1, CE2, CE3, CE4, CE5, and CE6), the preferred path for prefixes will be from eBGP, specifically from the R4 router. Although there might be paths from R5 and R6 routers and also from R1 and R2 routers through iBGP, the selection of best paths will prioritize eBGP multipaths from R4. This is the classic behavior. In classic eBGP, unequal-cost paths are employed, leading to the disregard of metrics. However, you rely on the IGP metric for optimal performance.

After Implementing This Feature

The iBGP paths with the shortest AS-PATH length are chosen for R5 and R6 router paths. The same iBGP multipath selection process applies to paths from R1 and R2 routers. As a result, the R1 and R2 routers establish an iBGP peering session with the R3 router. Therefore, a combination of eBGP and iBGP paths, referred to as eiBGP, is now available for prefixes advertised to hosts beyond the CE devices. The CE routers require load balancing of prefixes to R3 router and R4 router. However, it is necessary to exclude paths originating from R5 and R6 routers and R1 and R2 routers. Therefore, you must configure additive community on the R1 router and R2 routers towards the R5 and R6 routers.

With the setup depicted in the topology, you can establish the coexistence of both eBGP and iBGP, thus enabling seamless transition from utilizing eBGP multipaths to iBGP multipaths. By including the default VRF in policy-based multipath selection, you apply route policies to control how traffic is distributed within your network. By leveraging the BGP attributes such as BGP communities, nexthops, and path types within these route policies, you determine path selection. For example, you can use BGP communities to prioritize certain routes or manipulate nexthops to direct traffic over specific paths. This enables you to optimize routing decisions based on your specific requirements and goals, allowing you to gain control over traffic distribution and load-balancing capabilities across various BGP variations within your network.

By enabling ECMP, you allow a router to distribute traffic evenly across multiple equal-cost paths. This ensures that each path carries a portion of the traffic load, preventing any single path from becoming overwhelmed. By enabling the ECMP option in eiBGP, you allow the router to consider multiple iBGP paths with equal costs as viable options for traffic distribution. These paths are treated as equal-cost paths. This enhances load balancing in your network.

Benefits

This feature, with the inclusion of policy-based multipath selection, enables you to gain control over traffic distribution and load-balancing capabilities across various BGP variations, including iBGP, eBGP, and eiBGP. This is achieved through the utilization of BGP communities, nexthops, and path types.

Neglecting the utilization of BGP communities, nexthops, and path types within the default VRF during policy-based multipath selection can lead to limited control over traffic routing. The absence of BGP communities hinders the ability to apply specific policies to route updates, while ignoring nexthops and path types diminishes the accuracy of path selection decisions. This may result in suboptimal traffic distribution and load balancing.

Not applying ECMP within eiBGP can make the router to depend on its default path selection procedure to designate a singular optimal route from the accessible iBGP paths. This approach does not yield the load balancing and traffic distribution advantages offered by ECMP.

Restrictions for EIBGP Policy-Based Multipath with Equal Cost Multipath

The following are the restrictions:

- Configuring eiBGP along with either eBGP or iBGP is not allowed.
- The maximum-paths route policy allows for checks on community, nexthop, and path type only.
- The usage of the Accumulated Interior Gateway Protocol (AIGP) metric attribute is restricted only to equal-cost EIBGP scenarios.
- The OpenConfig model is not supported.
- When configuring eBGP and iBGP multipath together, it is possible to assign distinct or identical route policies to each of them. However, the selection of the policy to be applied between eBGP and iBGP is determined by the bestpath path type of the prefixes. If a prefix is determined to have a better path via

iBGP, the iBGP route policy will be applied, while for prefixes where eBGP is deemed better, the eBGP route policy will be applied.

Configure EIBGP Policy-Based Multipath with Equal Cost Multipath

Configuration Example

Perform the following steps to configure EIBGP Policy-Based Multipath with Equal Cost Multipath:

- Configure the community, path-type, or nexthop.
- Configure the route-policy with the multipath selection and equal-cost multipath for eiBGP.

Configure the community-set from the R1 and R2 routers

```
Router(config)# community-set ABC
Router(config-comm)# 2:1
Router(config-comm)# end-set
```

Configure the route-policy and equal-cost multipath option for eiBGP

The route-policy EIBGP is configured on R1 and R2 routers. This route-policy examines the BGP communities associated with BGP routes and takes specific actions based on the community values. If the community matches “ABC”, the route is not selected for multipath. For all the other cases, the router selects a path for multipath if it matches the best-path's metric and has the same path-type (i.e., iBGP or eBGP). If the path-type is different from the best path-type, it must be the best among the other path types. In addition to community, you also use path-type or next-hop as a route-policy option.

```
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
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
```

Running Configuration

```
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
  !
```


Verification

Verify that the router supports eiBGP multipath for this destination, and the route entries has been successfully received and processed.

```
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

```

ECMP Out of Resource Avoidance

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
ECMP Out of Resource Avoidance	Release 24.2.11	<p>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.</p> <p>The feature introduces these changes:</p> <p>CLI:</p> <ul style="list-style-type: none"> • prefix-ecmp-delay • cef load-balancing recursive oor mode dampening-and-dlb <p>YANG Data Models:</p> <ul style="list-style-type: none"> • 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 <p>(see GitHub, YANG Data Models Navigator)</p>

Cisco 8000 routers may encounter transient Equal-Cost Multi-Path (ECMP) resource shortages (Out of Resource condition) and subsequent traffic drops for IP-BGP routes under the following conditions:

- Data center migrations or network maintenance events, such as data center cost-in and cost-out.
- The introduction of new data center sites, which can lead to network instability and a temporary increase in ECMP resource usage.

After the network stabilizes, the router gracefully recovers from the ECMP spike. However, the traffic that was dropped during an OOR condition doesn't automatically recover.

Avoiding OOR Conditions

This feature allows the hardware resource usage to be tracked using an inline resource tracking mechanism within the Forwarding Information Base (FIB). Inline resource tracking provides real-time feedback on resource consumption directly within the FIB, which is beneficial for managing hardware resources more effectively. This approach allows for admission control mechanisms within the Border Gateway Protocol (BGP) and the FIB. These mechanisms can cache updates and delay certain operations until the OOR condition is resolved, ensuring that the system doesn't exceed its resource capacity.

When the resource utilization reaches a predefined threshold, BGP delays best path selection and route installation into the Routing Information Base (RIB), while the FIB delays hardware programming. This delay is configurable and is designed to prevent the system from reaching a state where it can't accommodate new routes because of resource constraints. By allowing BGP to delay route updates and FIB to delay hardware programming when resources are low, the system can avoid entering an OOR state, thereby achieving minimal to zero traffic loss and improved network performance.

FIB Dampening

When the resource usage reaches the configured dampening threshold, instead of immediately programming every route update into the hardware, the FIB consolidates or caches the route updates in the CPU memory, and delays the hardware programming. This approach prevents a sudden overload of the network's resources and keeps traffic flowing without interruption, even when resources are low.

FIB dampening is disabled by default. You can enable it through CEF configuration.

Dampening Switchover

Dampening Switchover is a mechanism that can detect when the state of route churn stabilizes. Once stability is detected, the route updates of stable state are programmed into the hardware.

Forced Switchover

If the network continues to experience churn and the dampening switchover algorithm couldn't find a stable state, a forced dampening switchover occurs once the maximum dampening duration expires. The default duration for this dampening period is typically set to 5 minutes.

During a forced switchover, some routes may be switched to Destination-Based Load Balancing (DLB) mode. This switch depends on the hardware resource usage. If the hardware resource usage exceeds the configured DLB threshold, the system may enter the DLB mode.

Destination-Based Load Balancing (DLB)

Routes are programmed in DLB mode only under specific conditions:

- **New Route Installation:** If a new route is being installed and the current hardware resource usage exceeds the configured DLB threshold, the route should be programmed in DLB mode to prevent an OOR condition.
- **Dampening Switchover:** During a forced dampening switchover, if the hardware resource usage is above the DLB threshold, the routes are programmed in DLB mode.

Uni-path Mode

DLB operates in a uni-path mode, which means that when DLB is triggered, the router selects a single path for forwarding traffic instead of multiple equal-cost paths. This is a protective measure to prevent the system from hitting an Out of Resource (OOR) condition.

Link-Over-Subscription Risk

When DLB mode is activated, the ability to spread data traffic evenly across multiple paths (ECMP) isn't available. This can lead to a risk of link over-subscription, as traffic that could have been distributed across several paths is now sent over a single path.

Automatic Switching Between DLB and ECMP

The system automatically switches between DLB and ECMP modes based on the current hardware resource utilization. If the hardware resource usage falls below the configured DLB threshold, the system reverts to using ECMP for the affected routes. Conversely, if the resource usage reaches the configured DLB threshold again, the system switches back to DLB mode.

Limitations for ECMP OOR Avoidance

These limitations apply to the ECMP OOR Avoidance feature:

- Resource accounting is designed only for deployments without MPLS in the path, such as IGP with MPLS, BGP LU/VPN, and so on. In cases where MPLS is present, and the system detects a significant number of Link Down Indications (LDIs) with MPLS protocol (more than approximately 1000 LDI), the system self-adjusts by increasing the resource count to account for the maximum MPLS paths. MPLS resource usage will only be increased after the system identifies considerable usage, to prevent misclassification of internal labels (like BFD internal label) as MPLS deployment.
- Resource accounting will only cover recursive and non-recursive LDI utilized by FIB. Other objects or features (for example, L2) that reserve ECMP or members will not be accounted for.
- The inline resource accounting in FIB may not align with the SDK resource accounting that is displayed in the **show controller npu resource** command output.
- FIB is not expected to transition LDIs from one load-balancing level to another (for example, SHLDI to REC_SHLDI or to PHLDI, and so on.). If any such transition occurs, the system disables resource monitoring accounting and triggers a warning message to alert the user. This precaution is necessary because different counters are used for different levels, and transitions could lead to inaccuracies in resource accounting.
- Resource accounting is not enabled for management interfaces and special (drop) adjacencies.

Configure BGP for ECMP OOR Avoidance

In BGP, you must configure the ECMP delay duration and the resource usage threshold limit.

Procedure

- Step 1** Execute the **prefix-ecmp-delay** *interval_value* **oor-threshold** *threshold_value* command to configure the ECMP delay duration and the OOR threshold value.

Example:

```
router bgp 100
  address-family ipv4 unicast
    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 2** Execute 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.

Example:

```
Router# show bgp ipv4 unicast process detail performance-statistics | b OOR
Fri Jun  7 17:35:20.284 UTC
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      ---
```

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 3** Execute the **show bgp location detail** command to view the details of BGP prefix delays.

```
Router# show bgp 209.165.201.9/27 detail
Wed Jul 31 14:01:13.358 EDT
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 highlighted content in the sample output indicates that the BGP prefix download to the RIB has been delayed.

Configure Dampening and DLB Modes

In FIB, you must enable dampening and DLB modes.

Procedure

- Step 1** To enable dampening and DLB features with their default values, use the **cef load-balancing recursive oor mode dampening-and-dlb** command.

Example:

```
Router(config)# cef load-balancing recursive oor mode dampening-and-dlb
```

The default hardware usage values for FIB dampening and DLB are 70% and 90% respectively. The default FIB dampening switchover duration is 300 seconds.

- a) To manually configure the FIB dampening switchover duration, use the **cef load-balancing recursive oor mode dampening-and-dlb max-duration value** command.

Example:

```
Router(config)# cef load-balancing recursive oor mode dampening-and-dlb max-duration 500
```

The FIB dampening switchover duration value ranges from 1 second to 600 seconds. FIB dampening and DLB are enabled with default hardware usage values (70%, and 90%).

- b) To manually configure the FIB dampening threshold value, FIB dampening maximum switchover duration, and DLB threshold value, use the **cef load-balancing recursive oor mode dampening-and-dlb dampening resource-threshold mbb_threshold max-duration value dlb resource-threshold dlb_threshold** command.

Example:

```
Router(config)# cef load-balancing recursive oor mode dampening-and-dlb dampening resource-threshold
80 max-duration 400 dlb resource-threshold 50
```

The FIB dampening threshold value ranges from 1 through 99, the FIB dampening switchover duration value ranges from 1 second to 600 seconds, and the DLB threshold value ranges from 1 through 99.

Step 2 When Hierarchical Load Balancing (HLB) routes are present, configure the **cef load-balancing mode hierarchical ecmp min-paths** *value* command.

Example:

```
Router(config)# cef load-balancing mode hierarchical ECMP min-paths 100
```

The minimum paths value ranges from 1 through 128.

Note

Before Release 24.2.1, the **cef hierarchical-load-balancing ecmp min-paths** *value* command was used to enable HLB with ECMP.

Step 3 You can always monitor the syslog messages to see if dampening or DLB is triggered. If the syslog messages are not displayed by default on the console, use the **show logging | i OOR** command to view the syslog messages.

Example:

```
Router#show logging | i OOR
Fri Jun 7 02:05:08.556 EDT
RP/0/RP0/CPU0:Jun 7 01:50:52.159 EDT: fib_mgr[408]: %ROUTING-FIB-4-LB_OOR_DAMPENING_HANDLING : Enter
Load Balancing OOR Dampening mode. HW resmon: 58%
LC/0/1/CPU0:Jun 7 01:50:52.159 EDT: fib_mgr[253]: %ROUTING-FIB-4-LB_OOR_DAMPENING_HANDLING : Enter
Load Balancing OOR Dampening mode. HW resmon: 58%
LC/0/6/CPU0:Jun 7 01:50:52.159 EDT: fib_mgr[265]: %ROUTING-FIB-4-LB_OOR_DAMPENING_HANDLING : Enter
Load Balancing OOR Dampening mode. HW resmon: 58%
RP/0/RP1/CPU0:Jun 7 01:50:52.158 EDT: fib_mgr[213]: %ROUTING-FIB-4-LB_OOR_DAMPENING_HANDLING : Enter
Load Balancing OOR Dampening mode. HW resmon: 58%
RP/0/RP1/CPU0:Jun 7 01:50:56.219 EDT: fib_mgr[213]: %ROUTING-FIB-4-LB_OOR_DLB_HANDLING : Enter Load
Balancing OOR DLB (uni-path) mode. HW resmon: 85%
RP/0/RP0/CPU0:Jun 7 01:50:56.220 EDT: fib_mgr[408]: %ROUTING-FIB-4-LB_OOR_DLB_HANDLING : Enter Load
Balancing OOR DLB (uni-path) mode. HW resmon: 85%
LC/0/6/CPU0:Jun 7 01:50:56.223 EDT: fib_mgr[265]: %ROUTING-FIB-4-LB_OOR_DLB_HANDLING : Enter Load
Balancing OOR DLB (uni-path) mode. HW resmon: 85%
LC/0/1/CPU0:Jun 7 01:50:56.224 EDT: fib_mgr[253]: %ROUTING-FIB-4-LB_OOR_DLB_HANDLING : Enter Load
Balancing OOR DLB (uni-path) mode. HW resmon: 85%
LC/0/6/CPU0:Jun 7 01:50:56.931 EDT: npu_drvr[296]: %PLATFORM-OFA-4-OOR_YELLOW : NPU 1, Table npu,
Resource stage_lb_member
RP/0/RP1/CPU0:Jun 7 01:55:56.357 EDT: fib_mgr[213]: %ROUTING-FIB-4-LB_OOR_DAMPENING_EXIT : Exit FIB
Load Balancing OOR Dampening. HW resmon: 85%
RP/0/RP0/CPU0:Jun 7 01:55:56.386 EDT: fib_mgr[408]: %ROUTING-FIB-4-LB_OOR_DAMPENING_EXIT : Exit FIB
Load Balancing OOR Dampening. HW resmon: 85%
LC/0/6/CPU0:Jun 7 01:55:56.888 EDT: fib_mgr[265]: %ROUTING-FIB-4-LB_OOR_DAMPENING_EXIT : Exit FIB
Load Balancing OOR Dampening. HW resmon: 85%
LC/0/1/CPU0:Jun 7 01:55:56.975 EDT: fib_mgr[253]: %ROUTING-FIB-4-LB_OOR_DAMPENING_EXIT : Exit FIB
Load Balancing OOR Dampening. HW resmon: 85%
LC/0/1/CPU0:Jun 7 02:04:10.037 EDT: fib_mgr[253]: %ROUTING-FIB-4-LB_OOR_DAMPENING_HANDLING : Enter
Load Balancing OOR Dampening mode. HW resmon: 84%
LC/0/6/CPU0:Jun 7 02:04:10.039 EDT: fib_mgr[265]: %ROUTING-FIB-4-LB_OOR_DAMPENING_HANDLING : Enter
Load Balancing OOR Dampening mode. HW resmon: 84%
RP/0/RP0/CPU0:Jun 7 02:04:10.048 EDT: fib_mgr[408]: %ROUTING-FIB-4-LB_OOR_DAMPENING_HANDLING : Enter
Load Balancing OOR Dampening mode. HW resmon: 84%
RP/0/RP1/CPU0:Jun 7 02:04:10.055 EDT: fib_mgr[213]: %ROUTING-FIB-4-LB_OOR_DAMPENING_HANDLING : Enter
Load Balancing OOR Dampening mode. HW resmon: 84%
```

This sample output shows the history of routes entering and exiting the dampening and DLB modes.

Step 4 To verify the hardware resource usage of the platform, run the **show cef misc** command.

Example:

IPv4

```
Router# show cef misc location 0/6/CPU0 | i LVL
Fri Jun  7 02:04:47.585 EDT
LVL1 LB Group:      Max: 8192; Used: 823(10%); high watermark: 1042, Jun  7 01:50:56.223 (LB OOR
threshold: Dampening,40%; DLB,85%);
LVL1 LB Member Paths: Max: 16384; Used: 7566(46%); high watermark: 13969, Jun  7 02:04:19.891 (LB
OOR threshold: Dampening,40%; DLB,85%);
LVL2 LB Group:      Max: 8192; Used: 3289(40%); high watermark: 3289, Jun  7 02:04:11.571;
LVL2 LB Member Paths: Max: 16384; Used: 4671(28%); high watermark: 4671, Jun  7 02:04:11.571
```

Example:

IPv6

```
Router# show cef ipv6 misc location 0/6/CPU0 | i LVL
Fri Jun  7 02:04:54.442 EDT
LVL1 LB Group:      Max: 8192; Used: 823(10%); high watermark: 1042, Jun  7 01:50:56.223 (LB OOR
threshold: Dampening,40%; DLB,85%);
LVL1 LB Member Paths: Max: 16384; Used: 7566(46%); high watermark: 13969, Jun  7 02:04:19.891 (LB
OOR threshold: Dampening,40%; DLB,85%);
LVL2 LB Group:      Max: 8192; Used: 3289(40%); high watermark: 3289, Jun  7 02:04:11.571;
LVL2 LB Member Paths: Max: 16384; Used: 4671(28%); high watermark: 4671, Jun  7 02:04:11.571
```

This example shows that the percentage of hardware resource used (46%) is greater than the configured dampening percentage (40%).

Note

Since IPv4 and IPv6 counters share the same resources, the hardware usage values in both IPv4 and IPv6 outputs are identical.

Step 5

To verify entries that are queued in the FIB OOR retry queue based on the object queue ID, use the **show cef object-queue queue-id queue_id** command.

Example:

```
Router# show cef object-queue queue-id 23 detail location 0/6/CPU0
Fri Jun  7 00:57:19.942 EDT
OBJ_PARTITION_MARKER id:PiDLB
  objs:0, walks:0, walked pl:0 route:0, active N, last-obj-add:Not Yet Recorded
  ptr: 0x308c152610
  obj type: OBJ_MARKER, flags: 0, refcnt: 0
  update time May 31 13:53:49.105
OBJ_PARTITION_MARKER id:MBBO
  objs:42, walks:0, walked pl:0, last-obj-add:Jun  7 00:57:14.996
  ptr: 0x308c152a90
  obj type: OBJ_MARKER, flags: 0, refcnt: 0
  update time May 31 13:53:49.105PATHLIST pl:0x3094a09f98 paths:50 pl-type:Shared refcnt:500
    1st prefix dependent: default 0xe0000000 209.1.83.1/32 leaf:0x309dbadfa8
  ptr: 0x308c3dadb40
  obj type: QUEUE-EXTENSION, flags: 0, refcnt: 0
  update time Jun  7 00:57:08.820
PATHLIST pl:0x3094a1de98 paths:54 pl-type:Shared refcnt:1500
  1st prefix dependent: default 0xe0000000 209.1.85.1/32 leaf:0x309dbcd3a8
  ptr: 0x308c3c87c8
  obj type: QUEUE-EXTENSION, flags: 0, refcnt: 0
  update time Jun  7 00:57:09.697
OBJ_PARTITION_MARKER id:MBBN
  objs:48, walks:7, walked pl:687 route:161479, merged-pl:17581, max-dur:300s, sleep:0, force:0, active
  Y, last-obj-add:Jun  7 00:57:14.994
  ptr: 0x308c152f10
  obj type: OBJ_MARKER, flags: 0, refcnt: 0
  update time May 31 13:53:49.103
  OOR Dampening - MBB Switchover History, num entries 7
```



```

-----
|      Time Stamp      | resource avail check (nhg/mem) | wlk-pl | pl-left | mbb2dlb | RM low/peak
|-----|
| Jun  1 18:09:18.592 | 155 / 5665 .vs. 59 / 2097 | 59      | 0       | 0       | 16% / 49%
|
| Jun  1 18:25:03.488 | 0   / 0     .vs. 371 / 3661 | 371     | 0       | 0       | 17% / 39%
|
| Jun  1 18:25:06.688 | 0   / 0     .vs. 23  / 1273 | 23      | 0       | 0       | 27% / 35%
|
| Jun  1 18:25:27.936 | 5   / 230   .vs. 62  / 3236 | 62      | 0       | 0       | 14% / 33%
|
| Jun  3 16:54:41.920 | 111 / 4970  .vs. 58  / 2119 | 58      | 0       | 0       | 23% / 51%
|
| Jun  3 18:47:12.128 | 79  / 4497  .vs. 46  / 1908 | 46      | 0       | 0       | 26% / 52%
|-----|
PATHLIST pl:0x3094a1bf98 paths:69 pl-type:Shared refcnt:1500
  1st prefix dependent: default 0xe0000000 209.1.85.1/32 leaf:0x309dbcd3a8
ptr: 0x308c3e3370
obj type: QUEUE-EXTENSION, flags: 0, refcnt: 0
update time Jun  7 00:57:08.817
PATHLIST pl:0x3094a1ff98 paths:61 pl-type:Shared refcnt:500
  1st prefix dependent: default 0xe0000000 209.1.83.1/32 leaf:0x309dbadfa8
ptr: 0x308c3d6f68
obj type: QUEUE-EXTENSION, flags: 0, refcnt: 0
update time Jun  7 00:57:08.567

```

This example indicates that the system is in dampening state.

MBBO (old path) has 54 paths, and **MBBN** (new path) has 69 paths.

PiDLB indicates that the prefix or route is programmed in uni-path to avoid ECMP OOR condition.

- a) To verify the event history of dampening switchover and DLB recovery, run the **show cef object-queue queue-id queue_id detail** command.

Example:

Dampening switchover

```

Router# show cef ipv6 object-queue queue-id 23 detail location 0/6/CPU0 | b MBB
Fri Jun  7 01:03:59.295 EDT
OBJ_PARTITION_MARKER id:MBBO
  objs:0, walks:0, walked pl:0, last-obj-add:Jun  7 00:56:47.889
ptr: 0x308cc88390
obj type: OBJ_MARKER, flags: 0, refcnt: 0
update time May 31 13:53:49.418
OBJ_PARTITION_MARKER id:MBBN
  objs:0, walks:7, walked pl:102 route:25251, merged-pl:162, max-dur:300s, sleep:0, force:1, active
N,
last-obj-add:Jun  7 00:56:56.796
ptr: 0x308cc88810
obj type: OBJ_MARKER, flags: 0, refcnt: 0
update time May 31 13:53:49.418
OOR Dampening - MBB Switchover History, num entries 7
-----
|      Time Stamp      | resource avail check (nhg/mem) | wlk-pl | pl-left | mbb2dlb | RM low/peak
|-----|
| May 31 22:24:51.840 | 53  / 229   .vs. 9   / 120   | 9       | 0       | 0       | 53% /
54% |
| May 31 22:25:43.296 | 0   / 0     .vs. 3   / 42    | 3       | 0       | 0       | 24% /
24% |
| Jun  3 16:53:30.624 | 227 / 1558  .vs. 24  / 325   | 24      | 0       | 0       | 37% /

```

```

45% |
| Jun  3 18:45:44.320 | 304 / 2246 .vs. 51 / 645 | 51 | 0 | 0 | 37% /
50% |
| Jun  3 18:46:34.496 | 0 / 0 .vs. 1 / 15 | 1 | 0 | 0 | 39% /
39% |
| Jun  3 18:47:12.128 | 1 / 13 .vs. 1 / 15 | 1 | 0 | 0 | 26% /
26% |
| Jun  7 01:01:55.840 | 0 / 0 .vs. 13 / 342 | 13 F | 0 | 12 | 46% /
48% |

```

```

-----
OBJ_PARTITION_MARKER id:MBBNR
objs:0, walks:1, walked pl:13, merged-pl:0, last-obj-add:Jun  7 00:56:56.796
ptr: 0x308cc88c90
obj type: OBJ_MARKER, flags: 0, refcnt: 0
update time May 31 13:53:49.418
OOR Dampening - HLB Site Routes MBB Switchover History, num entries 1
-----
|      Time Stamp      | wlk-pl | wlk-lf | pl-left | RM low/peak |
-----
| Jun  7 01:01:55.840 | 13  F  | 13     | 0       | 16% / 16% |
-----

```

```

OBJ_PARTITION_MARKER id:OOR
objs:0, walks:3, walked pl:10, last-obj-add:Jun  3 18:36:16.526
ptr: 0x308cc89110
obj type: OBJ_MARKER, flags: 0, refcnt: 0
update time May 31 13:53:49.418

```

In this sample output,

- Dampening switchover is configured with a dampening threshold of 300 s (5 mins). The objects remain in dampening queue for five minutes until the timer expires. After five minutes, the routes are programmed in ECMP mode or DLB mode based on the hardware resource state.
- **MBB Switchover History** displays the history of dampening switchovers happened at different time stamps.
 - **pl-left =0** implies an empty object queue.
 - **mbb2dlb =12** indicates that dampening switchover has happened and 12 routes will be programmed in DLB mode.
 - **F** indicates dampening switchover by force.
 - **active N** indicates that the system is not in dampening state.
- **HLB Site Routes MBB switchover history** displays the history of HLB site routes switchovers happened at different time stamps.

HLB routes use non recursive resources.

Example:

DLB recovery

```

Router# show cef object-queue queue-id 23 detail location 0/6/CPU0
Fri Jun  7 02:16:29.223 EDT
OBJ_PARTITION_MARKER id:PiDLB
objs:1536, walks:3, walked pl:3 route:37, active Y, last-obj-add:Jun  7 02:09:11.828
ptr: 0x308c152610
obj type: OBJ_MARKER, flags: 0, refcnt: 0
update time May 31 13:53:49.104
OOR Dampening - PI-DLB Recovery History, num entries 3
-----
|      Time Stamp      | wlk-pl | wlk-lf | pl-left | RM low/peak |
-----

```

```

-----
| Jun  7 02:04:08.832 | 1      | 16      | 1511     | 84% / 85% |
| Jun  7 02:04:11.008 | 1      | 18      | 1525     | 84% / 85% |
| Jun  7 02:04:20.096 | 1      | 3       | 1536     | 84% / 85% |
-----
PATHLIST pl:0x30a51b6698 paths:15 pl-type:Shared refcnt:10
  1st prefix dependent: default 0xe0000000 207.1.89.101/32 leaf:0x30a59daaa8
  ptr: 0x30a4fa1068
  obj type: QUEUE-EXTENSION, flags: 0, refcnt: 0
  update time Jun  7 01:50:56.233
PATHLIST pl:0x30a51b6798 paths:10 pl-type:Shared refcnt:9
  1st prefix dependent: default 0xe0000000 207.1.89.103/32 leaf:0x30a59daba8
  ptr: 0x30a4fa10f0
  obj type: QUEUE-EXTENSION, flags: 0, refcnt: 0
  update time Jun  7 01:50:56.233

```

In this sample output,

- **active Y** indicates that the DLB state is active.
- **PI-DLB Recovery History** displays the number of pathlists and leafs that are yet to be walked.
- The **objs** value and **pl-left** value will match most of the time.

Note

The object queue for line cards, for example, LC1, and LC2 can have similar or slightly different values.

Step 6 To verify if the route is installed in DLB mode, use the **show cef ipv4 / ipv6** command.

Example:

```

Router# show cef 209.165.200.225
Mon Nov 27 17:56:39.569 PST
198.0.0.2/32, version 12, PI-DLB, internal 0x1000001 0x0 (ptr 0x62f656d0) [1], 0x0 (0x0), 0x0 (0x0)
Updated Nov 27 17:55:40.203
Prefix Len 32, traffic index 0, precedence n/a, priority 0
gateway array (0x6323a8d0) reference count 2, flags 0x2010, source rib (7), 0 backups
  [1 type 3 flags 0x48449 (0x6329c0d8) ext 0x0 (0x0)]
LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
gateway array update type-time 1 Nov 27 17:55:40.203
via 10.0.0.2/32, 5 dependencies, recursive [flags 0x0]
  path-idx 0 NHID 0x0 [0x62f65cd8 0x0], Internal 0x643fc0a0
  next hop 10.0.0.2/32 via 10.0.0.2/32
via 11.0.0.2/32, 3 dependencies, recursive [flags 0x0]
  path-idx 1 NHID 0x0 [0x62f65a68 0x0], Internal 0x643fc1d0
  next hop 10.10.10.2/32 via 10.09.0.2/32

Load distribution: 0 (refcount 2)

Hash OK Interface Address
0 Y UNKNOWN intf 0x00000014 10.0.1.2

```

This sample output shows that the route is installed in DLB mode, and the single path is picked by Hash calculations.

Protection of Directly Connected EBGH Neighbors through Interface-Based LPTS Identifier

Table 11: Feature History Table

Feature Name	Release Name	Description
Protection of Directly Connected EBGH Neighbors through Interface-Based LPTS Identifier	Release 7.10.1	<p>We have enhanced the network security for directly connected eBGH neighbors by ensuring that only packets originating from designated eBGH 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.</p> <p>The feature introduces these changes:</p> <p>CLI:</p> <ul style="list-style-type: none"> • bgp lpts-secure-binding <p>YANG Data Model:</p> <ul style="list-style-type: none"> • Cisco-IOS-XR-um-router-bgp-cfg <p>(see GitHub, YANG Data Models Navigator)</p>

Local Packet Transport Services (LPTS) maintains tables describing all packet flows destined for the secure domain router (SDR), making sure that packets are delivered to their intended destinations.

With respect to BGP sessions, LPTS bindings can be categorized as follows:

- BGP Known: These LPTS entries correspond to BGP sessions with established neighbors.
- BGP Configured Peer: LPTS entries in this category are designated to receive the initial packets (TCP SYN and 3rd ACK) from specifically configured BGP neighbors.
- BGP Default Entries: This category encompasses LPTS entries that capture all packets originating from un-configured BGP neighbors.

An attacker who spoofs a packet using the exact combination of source IP, destination IP, source port, and destination port, and then floods these packets from another interface within the same VRF, will cause the packet to match the BGP known LPTS entry. As a result, the packet will traverse up to the TCP layer and potentially be dropped at that level. All BGP known LPTS entries share a common LPTS policer, which means that packets arriving through any of these entries will be policed at the specified rate.

However, if the attacker sends these packets at a rate exceeding the policer's defined rate, this will lead to congestion in this flow, adversely impacting BGP established peers. As a result, these BGP sessions may experience instability, which could lead to flapping.

This feature enables you to protect your network by adding an interface identifier for LPTS in directly connected eBGP neighbors. LPTS filters and polices the packets based on the type of flow rate you configure. This feature ensures that only packets originating from designated eBGP neighbors can traverse through a single interface, thus preventing IP spoofing. The interface identifier that is added will be passed to the LPTS and TCP only when the below-mentioned criteria are met:

- The BGP peer is configured to be external.
- The 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.
- The default eBGP TTL is used.
- The "ignore connected" option is not configured.
- A non-link local IPv6 neighbor address is configured.

In the LPTS binding process through the LPTS socket option, BGP generates a tuple for the interface identifier for every directly configured eBGP neighbor.

The configured BGP LPTS entry will only match an incoming connection (TCP SYN packet) if it is received from the programmed interface.

The BGP default entry handles incoming connections, or any other packets, received on interfaces other than the specified ones. These packets are subjected to rigorous policing and forwarded to TCP for reset generation. As a result, any spoofed packets arriving from non-desired interfaces will not affect the BGP configured peer LPTS entries.

Upon receiving a passive connection from the programmed interface and establishing it at the TCP level, TCP will inherit the same interface for the BGP known LPTS entry, which will be created for this specific connection.

Packets that match the source IP, destination IP, source port, destination port, and VRF information of an established connection, but are received from a different interface, will not be matched to the LPTS entry. As a result, these packets will be directed to the BGP default entry. This mechanism ensures that spoofed packets originating from non-desired interfaces will not affect the BGP known peer LPTS entries.

During the bind process for an active connection, BGP will also furnish the interface identifier. TCP will incorporate this interface information into the LPTS entry corresponding to the active connection, effectively safeguarding BGP known LPTS entries against spoofed packets that might match this connection but originate from a different interface.

Configure Protection of Directly Connected EBGP Neighbors through Interface-Based LPTS Identifier

To enable Local Packet Transport Services (LPTS) secure binding, perform the following steps:

```
Router# (config) router bgp 100
Router# (config-bgp) bgp lpts-secure-binding
```

Running Configuration

```
router bgp 100
  bgp lpts-secure-binding
```

Verification

Verify the LPTS bindings along with the connected interface identifier:

```
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
```

Verify that the LPTS secure binding is enabled:

```
Router# show bgp process | in LPTS
```

```
Wed Dec 14 14:28:33.779 PST
LPTS secure binding is enabled
```

Verify that the status of the connected interface identifier in LPTS is active:

```
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
```

Convergence for BGP Labeled Unicast PIC Edge

Table 12: Feature History Table

Feature Name	Release Information	Feature Description
Convergence for BGP Labeled Unicast PIC Edge	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *Convergence for BGP Labeled Unicast PIC Edge is now supported on Cisco 8712-MOD-M routers.
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.

BGP Labeled Unicast (LU) PIC Edge feature enables you to create and store both the primary and backup path in the Routing Information Base (RIB), Forwarding Information Base (FIB), and Cisco Express Forwarding. When the router detects a failure, the backup or alternate path immediately takes over, thus this feature enables fast failover and convergence in subseconds.

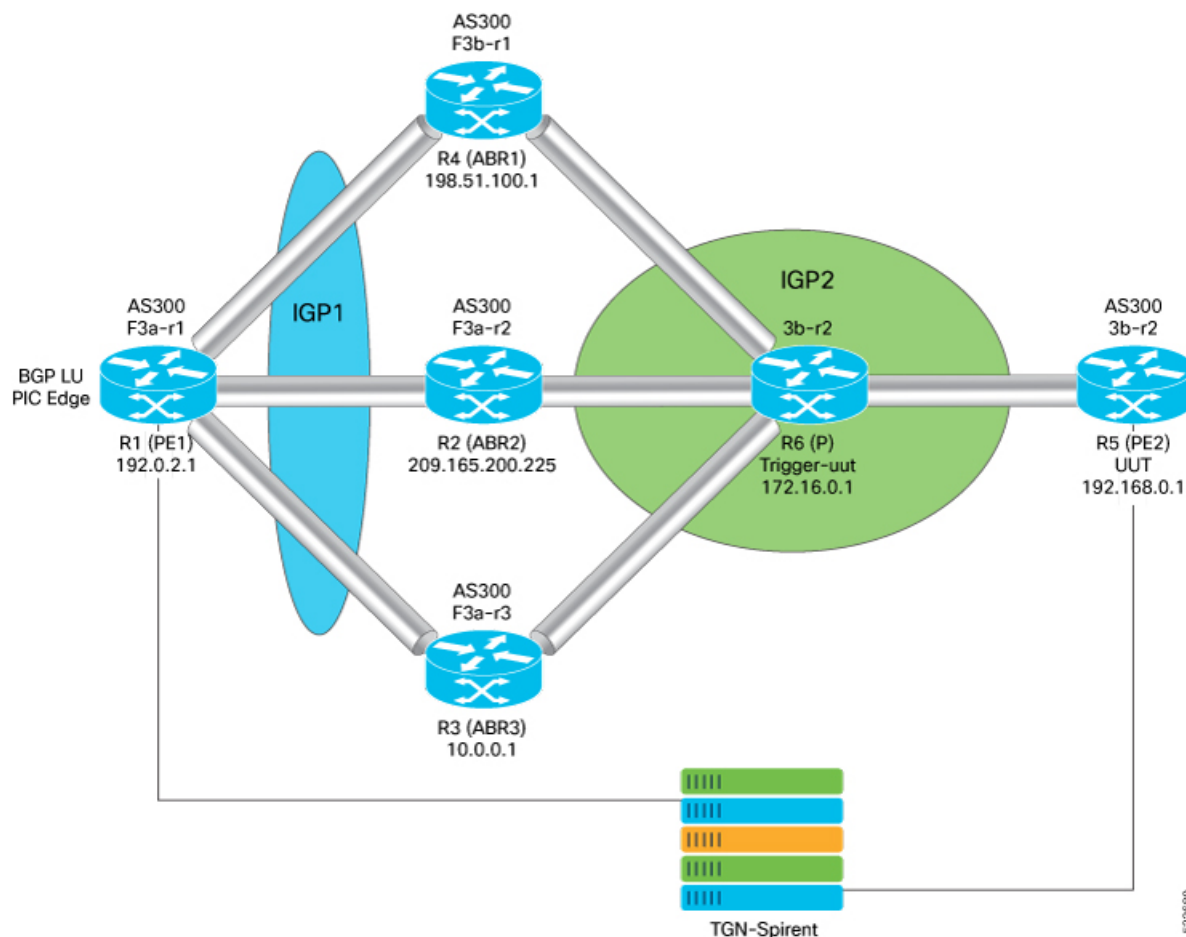
For BGP LU PIC Edge to work, the edge iBGP devices, such as ingress PEs and Autonomous System Border Router (ASBR), must support BGP PIC and must receive backup BGP next hop.

The topology diagram given below illustrates the Convergence for BGP Labeled Unicast PIC Edge feature. The topology is explained as follows:

- The BGP LU PIC Edge feature is enabled on a provider edge router, PE1.
- PE1 learns the BGP LU prefix from the remote PE router, PE2.
- PE1 routes traffic through the Area Border Routers, ABR1, ABR2 and ABR3. If one of them fails, the preprogrammed backup of the failed ABR routes the traffic.
- PE1 routes traffic through the Area Border Routers, ABR1, ABR2 and ABR3.
- PE2 is marked as the backup or alternate next hop and is programmed into the FIB of PE1.
- When PE1 learns PE2 is not reachable through ABR1, it immediately changes the BGP next hop for the PE1's prefix to ABR2.
- The switchover occurs in less than a second regardless of the number of prefixes.
- Subsecond convergence occurs although updates to multiple BGP prefixes are pending.

Topology

Figure 5: BGP LU PIC Edge



Guidelines and Limitations

This feature supports BGP multipaths that allows the router to install multiple internal BGP paths and multiple external BGP paths to the forwarding table. The multiple paths enable BGP to load balance traffic across multiple links.

The convergence time is independent of the BGP LU route scale.

Configure Convergence for BGP Labeled Unicast PIC Edge

Perform the following steps to configure Convergence for BGP Labeled Unicast PIC Edge:

- Configure BGP labeled unicast and attach route-policy to BGP address families.
- Configure BGP labeled unicast multipath and attach route-policy to BGP address families

```
Router(config)# route-policy BGP-PIC-EDGE
Router(config-rpl)# set path-selection backup 1 install
Router(config-rpl)# end-policy
```



```

Router(config)# end
Router(config)# router bgp 200
Router(config-bgp)# bgp router-id 10.0.0.1
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

/*Perform the following steps to configure BGP labeled unicast multipath and attach
route-policy to BGP address families: */
Router(config)# route-policy BGP-PIC-EDGE-MULTIPATH
Router(config-rpl)# set path-selection backup 1 install multipath-protect
Router(config)# end-policy
Router(config)# router bgp 200
Router(config)# bgp router-id 192.168.1.0
Router(config)# address-family ipv4 unicast
Router(config)# maximum-paths ibgp 2
Router(config)# additional-paths receive
Router(config)# additional-paths send
Router(config)# additional-paths selection route-policy BGP-PIC-EDGE-MULTIPATH

```

Running Configuration

```

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

```

Verification

Verify that the backup path is established.

```

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}

```

Black Box Monitoring

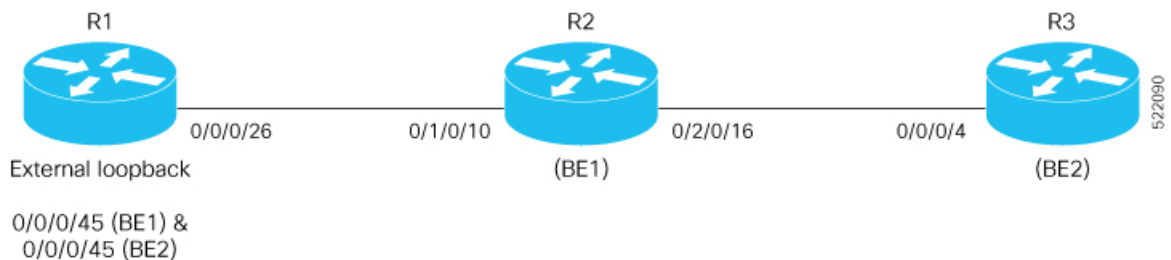
Table 13: Feature History Table

Feature Name	Release Information	Feature Description
Black Box Monitoring	Release 7.3.2	This feature enables you to set up forwarding path on the router that you can use to probe customer circuits for system metrics specific to the network devices. Such monitoring helps you to keep up the service level agreements with your customers.

This feature uses a technique whereby a dummy BGP session is established across the GRE encapsulation and decapsulation infrastructure. To terminate the dummy BGP session, the router peers to an address that is configured on the peering fabric which is peering to itself.

The router must peer to an address which is configured on the PF, peering to itself in essence. The only way to make this work is by plugging two interfaces into one another with a physical cable. After two interfaces are connected to one another place one of them into a VRF so that the BGP session is brought up. A router does not attempt to establish a BGP session to itself normally, so you must separate the routing table using a VRF. On the other interface it is a 'normal' interface in the global vrf with the same configuration that is typically on a PF peering interface.

Configuration Example



Perform the following steps to configure BGP and GRE tunnel..

```

/* Configure the Local Proxy ARP on the Bundle-Ether interfaces.*/
Router(config)# interface Bundle-Ether1.1
Router(config-if)# ipv4 address 10.1.1.1 255.255.255.240
Router(config-if)# local-proxy-arp
Router(config-if)# encapsulation dot1q 12
Router(config-if)# ipv4 access-group acl-aa ingress

Router(config-if)# exit
Router(config)# interface Bundle-Ether2.1
Router(config-if)# vrf aa
Router(config-if-vrf)# ipv4 address 10.1.1.2 255.255.255.240
Router(config-if-vrf)# local-proxy-arp
Router(config-if-vrf)# encapsulation dot1q 12

/* Configure a bundle on FortyGigE interfaces.*/

```

```
Router(config)# interface FortyGigE 0/0/0/46
Router(config-if)# bundle id 1 mode on
Router(config-if)# exit
Router(config)# interface FortyGigE0/0/0/47
Router(config-if)# bundle id 2 mode on

/* Configure the access list.*/
Router(config-if)# ipv4 access-list acl-aa
Router(config-if)# 1 permit icmp any host 10.1.1.1 echo-reply
Router(config-if)# 2 permit ipv4 any any nexthop1 ipv4 100.100.2.2
Router(config-if)# 10 permit tcp any eq bgp any
Router(config-if)# 20 permit tcp any any eq bgp

/* Configure BGP.*/
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 10.10.10.10
Router(config-bgp)# bgp log neighbor changes detail
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp)# maximum-paths ebgp 64
Router(config-bgp)# maximum-paths ibgp 64

/* Apply route policy. */
Router(config)# address-family vpnv4 unicast
Router(config-af)# vrf aa
Router(config-af)# rd auto
Router(config-af)# exitexit
Router(config)# address-family ipv4 unicast
Router(config)# exit
Router(config)# neighbor 10.1.1.1
Router(config-nbr)# remote-as 200
Router(config-nbr)# ebgp-multihop 4
Router(config-nbr)# exit
Router(config)# address-family ipv4 unicast
Router(config-af)# send-community ebgp
Router(config-af)# route-policy pass-all in
Router(config-af)# route-policy pass-all out

/* Configure loopback interfaces. */
Router(config)# interface Loopback1001
Router(config-if)# ipv4 address 10.10.10.10 255.255.255.255
Router(config)# exit
Router(config)# interface Loopback1002
Router(config-if)# vrf aa
Router(config-if-vrf)# ipv4 address 10.10.10.10 255.255.255.255

/* Configure a class map. */
Router(config)# class-map type traffic match-all aa
Router(config-cmap)# match protocol gre
Router(config-cmap)# match destination-address ipv4 10.10.10.10 255.255.255.255
Router(config-cmap)# end-class-map

/* Configure a policy map. */
Router(config)# policy-map type pbr pmap1
Router(config-pmap)# class type traffic aa
Router(config-pmap-c)# decapsulate gre
Router(config-pmap-c)# class type traffic class-default
Router(config-pmap-c)# end-policy-map

/* Configure VRF policy. */
Router(config)# vrf-policy
Router(config-vrf)# vrf default address-family ipv4 policy type pbr input pmap1
Router(config)# interface tunnel-ip 1100
Router(config-if)# ipv4 unnumbered Loopback1001
```

```
Router(config-if)#tunnel mode gre ipv4 encaps
Router(config-if)#tunnel source Loopback1001
Router(config-if)#tunnel destination 200.1.2.1
Router(config-if)#logging events link-status
```

Running Configuration

```
interface Bundle-Ether1.1
  ipv4 address 10.1.1.1 255.255.255.240
  local-proxy-arp
  encapsulation dot1q 12
  ipv4 access-group aa-acl ingress

interface Bundle-Ether2.1
  vrf aa
  ipv4 address 10.1.1.2 255.255.255.240
  local-proxy-arp
  encapsulation dot1q 12

interface FortyGigE0/0/0/46
  bundle id 1 mode on

interface FortyGigE0/0/0/47
  bundle id 2 mode on
  ipv4 access-list aa-acl
  1 permit icmp any host 10.1.1.1 echo-reply
  2 permit ipv4 any any nextthop1 ipv4 100.100.2.2
  10 permit tcp any eq bgp any
  20 permit tcp any any eq bgp

router bgp 100
  bgp router-id 10.10.10.10
  bgp log neighbor changes detail
  address-family ipv4 unicast
    maximum-paths ebgp 64
    maximum-paths ibgp 64
  !
  address-family vpnv4 unicast
  !
  vrf aa
    rd auto
    address-family ipv4 unicast
    !
    neighbor 10.1.1.1
      remote-as 200
      ebgp-multihop 4
    address-family ipv4 unicast
      send-community-ebgp
      route-policy pass-all in
      route-policy pass-all out

interface Loopback1001
  ipv4 address 10.10.10.10 255.255.255.255
RP/0/RP0/CPU0:SF-DD#sh run int loopback 1002
interface Loopback1002
  vrf aa
  ipv4 address 10.10.10.10 255.255.255.255

class-map type traffic match-all aa
  match protocol gre
  match destination-address ipv4 10.10.10.10 255.255.255.255
end-class-map
```

```

policy-map type pbr pmap1
  class type traffic aa
    decapsulate gre
  class type traffic class-default
end-policy-map
!
vrf-policy
  vrf default address-family ipv4 policy type pbr input pmap1

interface tunnel-ip1100
  ipv4 unnumbered Loopback1001
  tunnel mode gre ipv4 encap
  tunnel source Loopback1001
  tunnel destination 200.1.2.1
  logging events link-status

```

Verification

Verify the configuration of black box monitoring.

```

Router# show bgp vrf aa neighbors
BGP neighbor is 10.1.1.1, vrf aa
  Remote AS 200, local AS 100, external link
  Remote router ID 200.1.2.1
  BGP state = Established, up for 00:12:35
  NSR State: None
  Last read 00:00:30, 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 00:00:30, attempted 19, written 19
  Second last write 00:01:30, 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 Sep 29 05:50:49.983 last full not set pulse count 30
  Last write pulse rcvd before reset 00:00:00
Connections established 1; dropped 0
  Local host: 10.1.1.2, Local port: 52660, IF Handle: 0x00000000
  Foreign host: 10.1.1.1, Foreign port: 179
  Last reset 00:00:00
  External BGP neighbor may be up to 4 hops away.

```

BGP Labeled Unicast Version 6

Table 14: Feature History Table

Feature Name	Release Information	Feature Description

BGP Labeled Unicast Version 6	Release 7.3.16	<p>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.</p> <p>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.</p>
-------------------------------	----------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Overview of BGP Labeled Unicast

The BGP Labeled Unicast (LU) feature, also known as unified MPLS, provides MPLS transport between Provider Edge (PE) routers that are separated by either many IGP boundaries (intra-AS) or by many autonomous systems (inter-AS). Using autonomous systems border routers (ASBRs), you can advertise loopback prefixes of PEs and their MPLS label bindings: iBGP between area border routers (ABRs) and eBGP between autonomous system border routers. You can use Multihop eBGP between the PEs if they are in different autonomous systems (ASes) to exchange the VPN routes. You can run 6PE and other services between the PEs that have BGP LU connectivity.

The BGP LU feature lowers the IGP labeled prefix scale and adjacency scale values. If the router is not being configured with BGP LU, it is necessary to prevent lowering of scale values. Hence it is mandatory to configure the hw-module command before you enable the BGP LU feature. Restart the router for the hw-module command configuration to take effect.

The BGP Labeled Unicast Version 6 (BGP LU v6) feature extends the BGP Labeled Unicast (LU) functionality over IPv6.

Restrictions

- 6VPE over BGP LU feature is not supported.
- Inter-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 version 6 IPv6 unicast-address family is not supported.
- VPNV6 over BGP LU v6 is not supported.
- Link-local addresses are not supported.
- Rewrite cases, in which BGP LU is itself the transport, is not supported.
- Carrier Supporting Carrier Version 6 is not supported.
- Inter-AS Option-C with BGP LU Version 6 is not supported.

Configure BGP Labeled Unicast Version 6

```
Router(config)# hw-module profile cef bgplu enable
```

```

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
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
Router(config-bgp)# commit

```



Note Reload the router for the **hw-module profile cef bgplu enable** command to take effect.

Running Configuration

```

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

```

Verification

Verify that the BGP LU has been configured.

```

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

Verify the details of route paths along with the BGP and transport label information.

```

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
*/

```

Verify the BGP LU version 6 routes and BGP label information in BGP process.

```

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 Default Limits

Table 15: 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 the number of neighbors that can be configured on the router and on the maximum number of prefixes that are accepted from a peer for a given address family. This limitation safeguards the router from resource depletion caused by misconfiguration, either locally or on the remote neighbor. The following limits apply to BGP configurations:

- The default maximum number of peers that can be configured is 4000. The default can be changed using the **bgp maximum neighbor** command. The *limit* range is 1 to 15000. Any attempt to configure additional peers beyond the maximum limit or set the maximum limit to a number that is less than the number of peers currently configured will fail.
- To prevent a peer from flooding BGP with advertisements, a limit is placed on the number of prefixes that are accepted from a peer for each supported address family. The default limits can be overridden through configuration of the maximum-prefix *limit* command for the peer for the appropriate address family. The following default limits are used if the user does not configure the maximum number of prefixes for the address family:

- IPv4 Unicast: 1048576
- IPv4 Labeled-unicast: 131072
- IPv4 Tunnel: 1048576
- IPv6 Unicast: 524288
- IPv6 Labeled-unicast: 131072
- IPv4 Multicast: 131072
- IPv6 Multicast: 131072
- IPv4 MVPN: 2097152
- VPNv4 Unicast: 2097152
- IPv4 MDT: 131072
- VPNv6 Unicast: 1048576
- L2VPN EVPN: 2097152

A cease notification message is sent to the neighbor and the peering with the neighbor is terminated when the number of prefixes received from the peer for a given address family exceeds the maximum limit (either set by default or configured by the user) for that address family.

It is possible that the maximum number of prefixes for a neighbor for a given address family has been configured after the peering with the neighbor has been established and a certain number of prefixes have already been received from the neighbor for that address family. A cease notification message is sent to the neighbor and peering with the neighbor is terminated immediately after the configuration if the configured maximum number of prefixes is fewer than the number of prefixes that have already been received from the neighbor for the address family.

BGP Next Hop Tracking

Table 16: Feature History Table

Feature Name	Release Information	Feature Description
BGP Next Hop Tracking	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) * The BGP next hop tracking functionality is now extended to the Cisco 8712-MOD-M routers.

BGP Next Hop Tracking	Release 24.3.1	<p>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*)</p> <p>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.</p> <p>* The BGP next hop tracking functionality is now extended to:</p> <ul style="list-style-type: none"> • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
-----------------------	----------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

BGP receives notifications from the Routing Information Base (RIB) when next-hop information changes (event-driven notifications). BGP obtains next-hop information from the RIB to:

- Determine whether a next hop is reachable.
- Find the fully recursed IGP metric to the next hop (used in the best-path calculation).
- Validate the received next hops.
- Calculate the outgoing next hops.
- Verify the reachability and connectedness of neighbors.

BGP is notified when any of the following events occurs:

- Next hop becomes unreachable
- Next hop becomes reachable
- Fully recursed IGP metric to the next hop changes
- First hop IP address or first hop interface change
- Next hop becomes connected
- Next hop becomes unconnected
- Next hop becomes a local address
- Next hop becomes a nonlocal address



Note Reachability and recursed metric events trigger a best-path recalculation.

Event notifications from the RIB are classified as critical and noncritical. Notifications for critical and noncritical events are sent in separate batches. However, a noncritical event is sent along with the critical events if the noncritical event is pending and there is a request to read the critical events.

- Critical events are related to the reachability (reachable and unreachable), connectivity (connected and unconnected), and locality (local and nonlocal) of the next hops. Notifications for these events are not delayed.
- Noncritical events include only the IGP metric changes. These events are sent at an interval of 3 seconds. A metric change event is batched and sent 3 seconds after the last one was sent.

The next-hop trigger delay for critical and noncritical events can be configured to specify a minimum batching interval for critical and noncritical events using the **nexthop trigger-delay** command. The trigger delay is address family dependent.

The BGP next-hop tracking feature allows you to specify that BGP routes are resolved using only next hops whose routes have the following characteristics:

- To avoid the aggregate routes, the prefix length must be greater than a specified value.
- The source protocol must be from a selected list, ensuring that BGP routes are not used to resolve next hops that could lead to oscillation.

This route policy filtering is possible because RIB identifies the source protocol of route that resolved a next hop as well as the mask length associated with the route. The **nexthop route-policy** command is used to specify the route-policy.

Next Hop as the IPv6 Address of Peering Interface

BGP can carry IPv6 prefixes over an IPv4 session. The next hop for the IPv6 prefixes can be set through a nexthop policy. In the event that the policy is not configured, the nexthops are set as the IPv6 address of the peering interface (IPv6 neighbor interface or IPv6 update source interface, if any one of the interfaces is configured).

If the nexthop policy is not configured and neither the IPv6 neighbor interface nor the IPv6 update source interface is configured, the next hop is the IPv4 mapped IPv6 address.

IPv6 Multiprotocol BGP Peering Using a Global Address

When all ECMP links are shutdown except any one of the interfaces, the next-hop is changed from global address to link-local address which leads to traffic loss of all flows for a few seconds transient time.

You can then configure the **set next-hop ipv6-global** command under the BGP table-policy to avoid traffic loss over an undisturbed path.

BGP installs global ipv6 address nexthop for multipath routes and install *linklocal* and *ifhandle* for single path route to connect *ebgp neighbor* directly. You can configure the **set next-hop ipv6-global** command under the BGP table-policy as follows to set the global ipv6 address nexthop:

```
route-policy RESILIENT-HASH-V6
  if destination in (1000:1000::/32 le 128) or destination in (2000:1000::/32 le 128) then
    set load-balance ecmp-consistent
    set next-hop ipv6-global
  pass
endif
pass
end-policy
```

Scoped IPv4 Table Walk

To determine which address family to process, a next-hop notification is received by first de-referencing the gateway context associated with the next hop, then looking into the gateway context to determine which address families are using the gateway context. The IPv4 unicast address families share the same gateway context, because they are registered with the IPv4 unicast table in the RIB. As a result, the global IPv4 unicast table is processed when an IPv4 unicast next-hop notification is received from the RIB. A mask is maintained in the next hop, indicating the next hop belongs to IPv4 unicast. This scoped table walk localizes the processing in the appropriate address family table.

Reordered Address Family Processing

The software walks address family tables based on the numeric value of the address family. When a next-hop notification batch is received, the order of address family processing is reordered to the following order:

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

New Thread for Next-Hop Processing

The critical-event thread in the `spkr` process handles only next-hop, Bidirectional Forwarding Detection (BFD), and fast-external-failover (FEF) notifications. This critical-event thread ensures that BGP convergence is not adversely impacted by other events that may take a significant amount of time.

show, clear, and debug Commands

The **show bgp nexthops** command provides statistical information about next-hop notifications, the amount of time spent in processing those notifications, and details about each next hop registered with the RIB. The **clear bgp nexthop performance-statistics** command ensures that the cumulative statistics associated with the processing part of the next-hop **show** command can be cleared to help in monitoring. The **clear bgp nexthop registration** command performs an asynchronous registration of the next hop with the RIB.

The **debug bgp nexthop** command displays information on next-hop processing. The **out** keyword provides debug information only about BGP registration of next hops with RIB. The **in** keyword displays debug information about next-hop notifications received from RIB. The **out** keyword displays debug information about next-hop notifications sent to the RIB.

BGP Configuration

BGP in Cisco IOS XR software follows a neighbor-based configuration model that requires that all configurations for a particular neighbor be grouped in one place under the neighbor configuration. Peer groups are not supported for either sharing configuration between neighbors or for sharing update messages. The concept of peer group has been replaced by a set of configuration groups to be used as templates in BGP configuration and automatically generated update groups to share update messages between neighbors.

Configuration Modes

BGP configurations are grouped into modes. The following sections show how to enter some of the BGP configuration modes. From a mode, you can enter the **?** command to display the commands available in that mode.

Router Configuration Mode

The following example shows how to enter router configuration mode:

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

Router Address Family Configuration Mode

The following example shows how to enter router address family configuration mode:

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

Neighbor Configuration Mode

The following example shows how to enter neighbor configuration mode:

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

VRF Configuration Mode

The following example shows how to enter VPN routing and forwarding (VRF) configuration mode:

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

VRF Neighbor Configuration Mode

The following example shows how to enter VRF neighbor configuration mode:

```
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

The following example shows how to enter VRF neighbor address family configuration mode:

```
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 11.0.1.2
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

The following example shows how to enter VPNv6 address family configuration mode:

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

L2VPN Address Family Configuration Mode

The following example shows how to enter L2VPN address family configuration mode:

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

Neighbor Submode

Cisco IOS XR BGP uses a neighbor submode to make it possible to enter configurations without having to prefix every configuration with the **neighbor** keyword and the neighbor address:

- Cisco IOS XR software has a submode available for neighbors in which it is not necessary for every command to have a “neighbor x.x.x.x” prefix:

In Cisco IOS XR software, the configuration is as follows:

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

- An address family configuration submode inside the neighbor configuration submode is available for entering address family-specific neighbor configurations. In the Cisco IOS XR software, the configuration is as follows:

```
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
```

Configuration Templates

The **af-group**, **session-group**, and **neighbor-group** configuration commands provide template support for the neighbor configuration in Cisco IOS XR software.

The **af-group** command is used to group address family-specific neighbor commands within an IPv4, IPv6, address family. Neighbors that have the same address family configuration are able to use the address family

group (af-group) name for their address family-specific configuration. A neighbor inherits the configuration from an address family group by way of the **use** command. If a neighbor is configured to use an address family group, the neighbor (by default) inherits the entire configuration from the address family group. However, a neighbor does not inherit all of the configuration from the address family group if items are explicitly configured for the neighbor. The address family group configuration is entered under the BGP router configuration mode. The following example shows how to enter address family group configuration mode

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

The **session-group** command allows you to create a session group from which neighbors can inherit address family-independent configuration. A neighbor inherits the configuration from a session group by way of the **use** command. If a neighbor is configured to use a session group, the neighbor (by default) inherits the entire configuration of the session group. A neighbor does not inherit all of the configuration from a session group if a configuration is done directly on that neighbor. The following example shows how to enter session group configuration mode:

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

The **neighbor-group** command helps you apply the same configuration to one or more neighbors. Neighbor groups can include session groups and address family groups and can comprise the complete configuration for a neighbor. After a neighbor group is configured, a neighbor can inherit the configuration of the group using the **use** command. If a neighbor is configured to use a neighbor group, the neighbor inherits the entire BGP configuration of the neighbor group.

The following example shows how to enter neighbor group configuration mode:

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

The following example shows how to enter neighbor group address family configuration mode:

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

- However, a neighbor does not inherit all of the configuration from the neighbor group if items are explicitly configured for the neighbor. In addition, some part of the configuration of the neighbor group could be hidden if a session group or address family group was also being used.

Configuration grouping has the following effects in Cisco IOS XR software:

- Commands entered at the session group level define address family-independent commands (the same commands as in the neighbor submode).
- Commands entered at the address family group level define address family-dependent commands for a specified address family (the same commands as in the neighbor-address family configuration submode).

- Commands entered at the neighbor group level define address family-independent commands and address family-dependent commands for each address family (the same as all available **neighbor** commands), and define the **use** command for the address family group and session group commands.

Template Inheritance Rules

In Cisco IOS XR software, BGP neighbors or groups inherit configuration from other configuration groups.

For address family-independent configurations:

- Neighbors can inherit from session groups and neighbor groups.
- Neighbor groups can inherit from session groups and other neighbor groups.
- Session groups can inherit from other session groups.
- If a neighbor uses a session group and a neighbor group, the configurations in the session group are preferred over the global address family configurations in the neighbor group.

For address family-dependent configurations:

- Address family groups can inherit from other address family groups.
- Neighbor groups can inherit from address family groups and other neighbor groups.
- Neighbors can inherit from address family groups and neighbor groups.

Configuration group inheritance rules are numbered in order of precedence as follows:

1. If the item is configured directly on the neighbor, that value is used. In the example that follows, the advertisement interval is configured both on the neighbor group and neighbor configuration and the advertisement interval being used is from the neighbor configuration:

```
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
```

The following 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.
```

2. Otherwise, if an item is configured to be inherited from a session-group or neighbor-group and on the neighbor directly, then the configuration on the neighbor is used. If a neighbor is configured to be inherited from session-group or af-group, but no directly configured value, then the value in the session-group or af-group is used. In the example that follows, the advertisement interval is configured on a neighbor group and a session group and the advertisement interval value being used is from the session group:

```
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
```

The following 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.
```

3. Otherwise, if the neighbor uses a neighbor group and does not use a session group or address family group, the configuration value can be obtained from the neighbor group either directly or through inheritance. In the example that follows, the advertisement interval from the neighbor group is used because it is not configured directly on the neighbor and no session group is used:

```

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

```

The following 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.

```

To illustrate the same rule, the following example shows how to set the advertisement interval to 15 (from the session group) and 25 (from the neighbor group). The advertisement interval set in the session group overrides the one set in the neighbor group. The inbound policy is set to POLICY_1 from the neighbor group.

```

Routerconfig)# router bgp 140
Router(config-bgp)# session-group ADV
Router(config-bgp-sngrp)# advertisement-interval 15
Router(config-bgp-sngrp)# exit
Router(config-bgp)# neighbor-group ADV_2
Router(config-bgp-nbrgrp)# advertisement-interval 25
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
Router(config-bgp-nbrgrp-af)# route-policy POLICY_1 in
Router(config-bgp-nbrgrp-af)# exit
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# exit
Router(config-bgp)# neighbor 192.168.2.2
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# use session-group ADV

```

```
Router(config-bgp-nbr)# use neighbor-group ADV_2
```

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

```
Router# show bgp neighbors 192.168.2.2

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 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 00:02:03, due to BGP neighbor initialized
External BGP neighbor not directly connected.
```

4. Otherwise, the default value is used. In the example that follows, neighbor 10.0.101.5 has the minimum time between advertisement runs set to 30 seconds (default) because the neighbor is not configured to use the neighbor configuration or the neighbor group configuration:

```
Router(config)# router bgp 140
Router(config-bgp)# neighbor-group AS_1
Router(config-bgp-nbrgrp)# remote-as 1
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# neighbor-group adv_15
Router(config-bgp-nbrgrp)# remote-as 10
Router(config-bgp-nbrgrp)# advertisement-interval 15
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# neighbor 10.0.101.5
Router(config-bgp-nbr)# use neighbor-group AS_1
Router(config-bgp-nbr)# exit
Router(config-bgp)# neighbor 10.0.101.10
Router(config-bgp-nbr)# use neighbor-group adv_15
```

The following output from the **show bgp neighbors** command shows that the advertisement interval used is 30 seconds:

```
Router# show bgp neighbors 10.0.101.5

BGP neighbor is 10.0.101.5, 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 30 seconds
```

```

For Address Family: IPv4 Unicast
BGP neighbor version 0
Update group: 0.2
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 00:00:25, due to BGP neighbor initialized
External BGP neighbor not directly connected.

```

The inheritance rules used when groups are inheriting configuration from other groups are the same as the rules given for neighbors inheriting from groups.

Viewing Inherited Configurations

You can use the following **show** commands to view BGP inherited configurations:

show bgp neighbors

Use the **show bgp neighbors** command to display information about the BGP configuration for neighbors.

- Use the **configuration** keyword to display the effective configuration for the neighbor, including any settings that have been inherited from session groups, neighbor groups, or address family groups used by this neighbor.
- Use the **inheritance** keyword to display the session groups, neighbor groups, and address family groups from which this neighbor is capable of inheriting configuration.

The **show bgp neighbors** command examples that follow are based on this sample 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

```

show bgp neighbors

Use the **show bgp neighbors** command to display information about the BGP configuration for neighbors.

- Use the **configuration** keyword to display the effective configuration for the neighbor, including any settings that have been inherited from session groups, neighbor groups, or address family groups used by this neighbor.
- Use the **inheritance** keyword to display the session groups, neighbor groups, and address family groups from which this neighbor is capable of inheriting configuration.

The **show bgp neighbors** command examples that follow are based on this sample 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
```

show bgp af-group

Use the **show bgp af-group** command to display address family groups:

- Use the **configuration** keyword to display the effective configuration for the address family group, including any settings that have been inherited from address family groups used by this address family group.
- Use the **inheritance** keyword to display the address family groups from which this address family group is capable of inheriting configuration.
- Use the **users** keyword to display the neighbors, neighbor groups, and address family groups that inherit configuration from this address family group.

The **show bgp af-group** sample commands that follow are based on this sample 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
```

show bgp session-group

```

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

```

The following example displays sample output from the **show bgp af-group** command using the **configuration** keyword. This example shows from where each configuration item was inherited. The **default-originate** command was configured directly on this address family group (indicated by []). The **remove-private-as** command was inherited from address family group GROUP_2, which in turn inherited from address family group GROUP_3:

```

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]

```

The following example displays sample output from the **show bgp af-group** command using the **users** keyword:

```

Router# show bgp af-group GROUP_2 users

IPv4 Unicast: a:GROUP_1

```

The following example displays sample output from the **show bgp af-group** command using the **inheritance** keyword. This shows that the specified address family group GROUP_1 directly uses the GROUP_2 address family group, which in turn uses the GROUP_3 address family group:

```

Router# show bgp af-group GROUP_1 inheritance

IPv4 Unicast: a:GROUP_2 a:GROUP_3

```

show bgp session-group

Use the **show bgp session-group** command to display session groups:

- Use the **configuration** keyword to display the effective configuration for the session group, including any settings that have been inherited from session groups used by this session group.
- Use the **inheritance** keyword to display the session groups from which this session group is capable of inheriting configuration.
- Use the **users** keyword to display the session groups, neighbor groups, and neighbors that inherit configuration from this session group.

The output from the **show bgp session-group** command is based on the following 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
```

The following is sample output from the **show bgp session-group** command with the **configuration** keyword in session group configuration mode:

```
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]
```

The following is sample output from the **show bgp session-group** command with the **inheritance** keyword showing that the GROUP_1 session group inherits session parameters from the GROUP_3 and GROUP_2 session groups:

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

Session: s:GROUP_2 s:GROUP_3
```

The following is sample output from the **show bgp session-group** command with the **users** keyword showing that both the GROUP_1 and GROUP_2 session groups inherit session parameters from the GROUP_3 session group:

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

Session: s:GROUP_1 s:GROUP_2
```

show bgp session-group

Use the **show bgp session-group** command to display session groups:

- Use the **configuration** keyword to display the effective configuration for the session group, including any settings that have been inherited from session groups used by this session group.
- Use the **inheritance** keyword to display the session groups from which this session group is capable of inheriting configuration.
- Use the **users** keyword to display the session groups, neighbor groups, and neighbors that inherit configuration from this session group.

The output from the **show bgp session-group** command is based on the following session group configuration:

show bgp neighbor-group

```

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

```

The following is sample output from the **show bgp session-group** command with the **configuration** keyword in session group configuration mode:

```

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]

```

The following is sample output from the **show bgp session-group** command with the **inheritance** keyword showing that the GROUP_1 session group inherits session parameters from the GROUP_3 and GROUP_2 session groups:

```

Router# show bgp session-group GROUP_1 inheritance

Session: s:GROUP_2 s:GROUP_3

```

The following is sample output from the **show bgp session-group** command with the **users** keyword showing that both the GROUP_1 and GROUP_2 session groups inherit session parameters from the GROUP_3 session group:

```

Router# show bgp session-group GROUP_3 users

Session: s:GROUP_1 s:GROUP_2

```

show bgp neighbor-group

Use the **show bgp neighbor-group** command to display neighbor groups:

- Use the **configuration** keyword to display the effective configuration for the neighbor group, including any settings that have been inherited from neighbor groups used by this neighbor group.
- Use the **inheritance** keyword to display the address family groups, session groups, and neighbor groups from which this neighbor group is capable of inheriting configuration.
- Use the **users** keyword to display the neighbors and neighbor groups that inherit configuration from this neighbor group.

The examples are based on the following 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
Router(config-bgp-nbrgrp-af)# use af-group GROUP_2
Router(config-bgp-nbrgrp-af)# weight 100

```

The following is sample output from the **show bgp neighbor-group** command with the **configuration** keyword. The configuration setting source is shown to the right of each command. In the output shown previously, 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:

```

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]

```

The following is sample output from the **show bgp neighbor-group** command with the **inheritance** keyword. This output shows that the specified neighbor group GROUP_1 inherits session (address family-independent) configuration parameters from neighbor group GROUP_2. Neighbor group GROUP_2 inherits its session parameters from session group GROUP_3. It also shows that the GROUP_1 neighbor group inherits IPv4 unicast configuration parameters from the GROUP_2 neighbor group, which in turn inherits them from the GROUP_2 address family group, which itself inherits them from the GROUP_3 address family group:

```

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

```

The following is sample output from the **show bgp neighbor-group** command with the **users** keyword. This output shows that the GROUP_1 neighbor group inherits session (address family-independent) configuration parameters from the GROUP_2 neighbor group. The GROUP_1 neighbor group also inherits IPv4 unicast configuration parameters from the GROUP_2 neighbor group:

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

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

No Default Address Family

BGP does not support the concept of a default address family. An address family must be explicitly configured under the BGP router configuration for the address family to be activated in BGP. Similarly, an address family must be explicitly configured under a neighbor for the BGP session to be activated under that address family. It is not required to have any address family configured under the BGP router configuration level for a neighbor to be configured. However, it is a requirement to have an address family configured at the BGP router configuration level for the address family to be configured under a neighbor.

Neighbor Address Family Combinations

For default VRF, both IPv4 Unicast and IPv4 Labeled-unicast address families are supported under the same neighbor.

For non-default VRF, both IPv4 Unicast and IPv4 Labeled-unicast address families are not supported under the same neighbor. However, the configuration is accepted on the router with the following error:

```
bgp[1051]: %ROUTING-BGP-4-INCOMPATIBLE_AFI : IPv4 Unicast and IPv4 Labeled-unicast Address
families together are not supported under the same neighbor.
```

When one BGP session has both IPv4 unicast and IPv4 labeled-unicast AFI/SAF, then the routing behavior is nondeterministic. Therefore, the prefixes may not be correctly advertised. Incorrect prefix advertisement results in reachability issues. In order to avoid such reachability issues, you must explicitly configure a route policy to advertise prefixes either through IPv4 unicast or through IPv4 labeled-unicast address families.

Routing Policy Enforcement

External BGP (eBGP) neighbors must have an inbound and outbound policy configured. If no policy is configured, no routes are accepted from the neighbor, nor are any routes advertised to it. This added security measure ensures that routes cannot accidentally be accepted or advertised in the case of a configuration omission error.



Note This enforcement affects only eBGP neighbors (neighbors in a different autonomous system than this router). For internal BGP (iBGP) neighbors (neighbors in the same autonomous system), all routes are accepted or advertised if there is no policy.

Table Policy

The table policy feature in BGP allows you to configure traffic index values on routes as they are installed in the global routing table. This feature is enabled using the **table-policy** command and supports the BGP policy accounting feature.

BGP policy accounting uses traffic indices that are set on BGP routes to track various counters.

Table policy also provides the ability to drop routes from the RIB based on match criteria. This feature can be useful in certain applications and should be used with caution as it can easily create a routing ‘black hole’ where BGP advertises routes to neighbors that BGP does not install in its global routing table and forwarding table.

BGP Update Group

When a change to the configuration occurs, the router automatically recalculates update group memberships and applies the changes.

For the best optimization of BGP update group generation, we recommend that the network operator keeps outbound routing policy the same for neighbors that have similar outbound policies. This feature contains commands for monitoring BGP update groups.

BGP Update Generation and Update Groups

The BGP Update Groups feature separates BGP update generation from neighbor configuration. The BGP Update Groups feature introduces an algorithm that dynamically calculates BGP update group membership based on outbound routing policies. This feature does not require any configuration by the network operator. Update group-based message generation occurs automatically and independently.

BGP Cost Community

The BGP cost community is a nontransitive extended community attribute that is passed to internal BGP (iBGP) and confederation peers but not to external BGP (eBGP) peers. The cost community feature allows you to customize the local route preference and influence the best-path selection process by assigning cost values to specific routes. The extended community format defines generic points of insertion (POI) that influence the best-path decision at different points in the best-path algorithm.

How BGP Cost Community Influences the Best Path Selection Process

The cost community attribute influences the BGP best-path selection process at the point of insertion (POI). By default, the POI follows the Interior Gateway Protocol (IGP) metric comparison. When BGP receives multiple paths to the same destination, it uses the best-path selection process to determine which path is the best path. BGP automatically makes the decision and installs the best path in the routing table. The POI allows you to assign a preference to a specific path when multiple equal cost paths are available. If the POI is not valid for local best-path selection, the cost community attribute is silently ignored.

Cost communities are sorted first by POI then by community ID. Multiple paths can be configured with the cost community attribute for the same POI. The path with the lowest cost community ID is considered first. In other words, all cost community paths for a specific POI are considered, starting with the one with the lowest cost community. Paths that do not contain the cost community cost (for the POI and community ID being evaluated) are assigned the default community cost value (2147483647). If the cost community values are equal, then cost community comparison proceeds to the next lowest community ID for this POI.

To select the path with the lower cost community, simultaneously walk through the cost communities of both paths. This is done by maintaining two pointers to the cost community chain, one for each path, and advancing both pointers to the next applicable cost community at each step of the walk for the given POI, in order of community ID, and stop when a best path is chosen or the comparison is a tie. At each step of the walk, the following checks are done:

```
If neither pointer refers to a cost community,
    Declare a tie;

Elseif a cost community is found for one path but not for the other,
    Choose the path with cost community as best path;
Elseif the Community ID from one path is less than the other,
    Choose the path with the lesser Community ID as best path;
Elseif the Cost from one path is less than the other,
    Choose the path with the lesser Cost as best path;
Else Continue.
```



Note Paths that are not configured with the cost community attribute are considered by the best-path selection process to have the default cost value (half of the maximum value [4294967295] or 2147483647).

Applying the cost community attribute at the POI allows you to assign a value to a path originated or learned by a peer in any part of the local autonomous system or confederation. The cost community can be used as a “tie breaker” during the best-path selection process. Multiple instances of the cost community can be configured for separate equal cost paths within the same autonomous system or confederation. For example, a lower cost community value can be applied to a specific exit path in a network with multiple equal cost exit points, and the specific exit path is preferred by the BGP best-path selection process. .



Note The cost community comparison in BGP is enabled by default. Use the **bgp bestpath cost-community ignore** command to disable the comparison.

Cost Community Support for Aggregate Routes and Multipaths

The BGP cost community feature supports aggregate routes and multipaths. The cost community attribute can be applied to either type of route. The cost community attribute is passed to the aggregate or multipath route from component routes that carry the cost community attribute. Only unique IDs are passed, and only the highest cost of any individual component route is applied to the aggregate for each ID. If multiple component routes contain the same ID, the highest configured cost is applied to the route. For example, the following two component routes are configured with the cost community attribute using an inbound route policy:

- 10.0.0.1
 - POI=IGP
 - cost community ID=1
 - cost number=100
- 192.168.0.1
 - POI=IGP

- cost community ID=1
- cost number=200

If these component routes are aggregated or configured as a multipath, the cost value 200 is advertised, because it has the highest cost.

If one or more component routes do not carry the cost community attribute or the component routes are configured with different IDs, then the default value (2147483647) is advertised for the aggregate or multipath route. For example, the following three component routes are configured with the cost community attribute using an inbound route policy. However, the component routes are configured with two different IDs.

- 10.0.0.1
 - POI=IGP
 - cost community ID=1
 - cost number=100
- 172.16.0.1
 - POI=IGP
 - cost community ID=2
 - cost number=100
- 192.168.0.1
 - POI=IGP
 - cost community ID=1
 - cost number=200

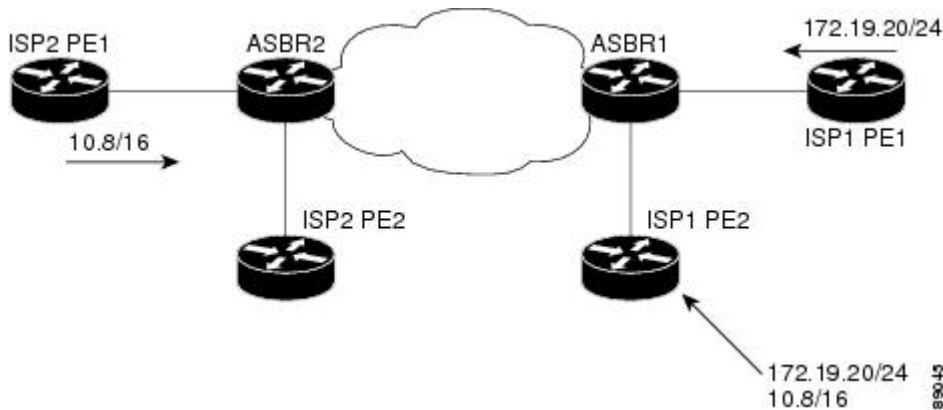
The single advertised path includes the aggregate cost communities as follows:

{POI=IGP, ID=1, Cost=2147483647} {POI=IGP, ID=2, Cost=2147483647}

Influencing Route Preference in a Multiexit IGP Network

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 6: Multiexit Point IGP Network



Both paths are considered to be equal by BGP. If multipath loadsharing is configured, both paths to the routing table are installed and are used to balance the load of traffic. If multipath load balancing is not configured, the BGP selects the path that was learned first as the best path and installs this path to the routing table. This behavior may not be desirable under some conditions. For example, the path is learned from ISP1 PE2 first, but the link between ISP1 PE2 and ASBR1 is a low-speed link.

The configuration of the cost community attribute can be used to influence the BGP best-path selection process by applying a lower-cost community value to the path learned by ASBR2. For example, the following configuration is applied to ASBR2:

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

The preceding route policy applies a cost community number of 1 to the 10.8.0.0 route. By default, the path learned from ASBR1 is assigned a cost community number of 2147483647. Because the path learned from ASBR2 has a lower-cost community number, the path is preferred.

Adding Routes to the Routing Information Base

If a nonsourced path becomes the best path after the best-path calculation, BGP adds the route to the Routing Information Base (RIB) and passes the cost communities along with the other IGP extended communities.

When a route with paths is added to the RIB by a protocol, RIB checks the current best paths for the route and the added paths for cost extended communities. If cost-extended communities are found, the RIB compares the set of cost communities. If the comparison does not result in a tie, the appropriate best path is chosen. If the comparison results in a tie, the RIB proceeds with the remaining steps of the best-path algorithm. If a cost community is not present in either the current best paths or added paths, then the RIB continues with the remaining steps of the best-path algorithm.

BGP DMZ Aggregate Bandwidth

Table 17: 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.
------------------------------------------------------------	---------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

BGP supports aggregating *dmz-link bandwidth* values of external BGP (eBGP) multipaths when advertising the route to interior BGP (iBGP) peer.

There is no explicit command to aggregate bandwidth. The bandwidth is aggregated if following conditions are met:

- The network has multipaths and all the multipaths have link-bandwidth values.
- The next-hop attribute set to next-hop-self. The next-hop attribute for all routes advertised to the specified neighbor to the address of the local router.
- There is no out-bound policy configured that might change the dmz-link bandwidth value.
- If the *dmz-link bandwidth* value is not known for any one of the multipaths (eBGP or iBGP), the *dmz-link* value for all multipaths including the best path is not downloaded to routing information base (RIB).
- The *dmz-link bandwidth* value of iBGP multipath is not considered during aggregation.
- The route that is advertised with aggregate value can be best path or add-path.
- Add-path does not qualify for DMZ link bandwidth aggregation as next hop is preserved. Configuring next-hop-self for add-path is not supported.
- For VPNv4 and VPNv6 afi, if *dmz link-bandwidth* value is configured using outbound route-policy, specify the route table or use the **additive** keyword. Else, this will lead to routes not imported on the receiving end of the peer.

```

extcommunity-set bandwidth dmz_ext
  1:8000
end-set
!
route-policy dmz_rp_vpn
  set extcommunity bandwidth dmz_ext additive      <<< 'additive' keyword.
  pass
end-policy

```

Removal of Link-Bandwidth Extended Community to iBGP Peers

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. The Removal of Link-Bandwidth Extended Community to iBGP Peers feature provides the flexibility to remove the DMZ link-bandwidth community to minimize the risk of exposure of the community parameters to networks zones where they are not recognized or unnecessary.

Configuration Example

Perform the following steps to allow users to be able to configure route-policy to remove the extended communities.

```
/* Delete all the 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

/* Delete only the extended communities that match an extended community mentioned in the
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

/* Delete all the extended communities. */
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
```

Verification

Verify the configuration that allows the user to remove a particular extended community.

```
Router# show bgp 10.9.9.9/32
Fri Aug 27 13:15:05.833 EDT
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

```

Configuring BGP DMZ Aggregate Bandwidth: Example

This is a sample configuration for Border Gateway Protocol Demilitarized Zone (BGP DMZ) link bandwidth. Consider the topology, R1---(iBGP)---R2---(iBGP)---R3:

1. On R1:

```

bgp: prefix p/n has:
path 1(bestpath)          with LB value 100
path 2(ebgp multipath)    with LB value 30
path 3(ebgp multipath)    with LB value 50

```

When best path is advertised to R2, send aggregated dmz-link bandwidth value of 180; aggregated value of paths 1, 2 and 3.

2. On R2:

```

bgp: prefix p/n has:
path 1(bestpath)          with LB value 60
path 2(ebgp multipath)    with LB value 200
path 3(ebgp multipath)    with LB value 50

```

When best path is advertised to R3, send aggregated dmz-link bandwidth value of 310; aggregated value of paths 1, 2 and 3.

3. On R3:

```

bgp: prefix p/n has:
path 1(bestpath)          with LB 180 {learned from R1}
path 2(ibgp multipath)    with LB 310 {learned from R2}

```

Configuring Policy-based Link Bandwidth: Example

This is a sample configuration for policy-based DMZ link bandwidth. The link-bandwidth ext-community can be set on a *per-path* basis either at the neighbor-in or neighbor-out policy attach-points. The *dmz-link-bandwidth* knob is configured under eBGP neighbor configuration mode. All paths received from that particular neighbor will be marked with the link-bandwidth extended community when sent to iBGP peers.

1. Configure inbound or outbound route-policy.

```

extcommunity-set bandwidth dmz_ext
  1:1290400000
end-set
!
route-policy dmz_rp
  set extcommunity bandwidth dmz_ext
  pass
end-policy
!

neighbor 10.0.101.1
  remote-as 1001
  address-family ipv4 unicast
    route-policy dmz_rp in          <<< Inbound route-policy.

```

```
route-policy pass out
!
```

2. Configure *dmz-link-bandwidth* under BGP neighbor.

```
neighbor 10.0.101.2
remote-as 1001
dmz-link-bandwidth <<< Under neighbor.
address-family ipv4 unicast
route-policy pass in
route-policy pass out
!
```

64-ECMP Support for BGP

IOS XR supports configuration of up to 64 equal cost multipath (ECMP) next hops for BGP. 64-ECMP is required in networks, where overloaded routers can load balance the traffic over as many as 64 LSPs.

BGP Best Path Algorithm

BGP routers typically receive multiple paths to the same destination. The BGP best-path algorithm determines the best path to install in the IP routing table and to use for forwarding traffic. This section describes the Cisco IOS XR software implementation of BGP best-path algorithm, as specified in Section 9.1 of the Internet Engineering Task Force (IETF) Network Working Group draft-ietf-idr-bgp4-24.txt document.

The BGP best-path algorithm implementation is in three parts:

- Part 1—Compares two paths to determine which is better.
- Part 2—Iterates over all paths and determines which order to compare the paths to select the overall best path.
- Part 3—Determines whether the old and new best paths differ enough so that the new best path should be used.



Note

The order of comparison determined by Part 2 is important because the comparison operation is not transitive; that is, if three paths, A, B, and C exist, such that when A and B are compared, A is better, and when B and C are compared, B is better, it is not necessarily the case that when A and C are compared, A is better. This nontransitivity arises because the multi exit discriminator (MED) is compared only among paths from the same neighboring autonomous system (AS) and not among all paths.

Comparing Pairs of Paths

Perform the following steps to compare two paths and determine the better path:

1. If either path is invalid (for example, a path has the maximum possible MED value or it has an unreachable next hop), then the other path is chosen (provided that the path is valid).
2. If the paths have unequal pre-bestpath cost communities, the path with the lower pre-bestpath cost community is selected as the best path.
3. If the paths have unequal weights, the path with the highest weight is chosen.



Note The weight is entirely local to the router, and can be set with the **weight** command or using a routing policy.

4. If the paths have unequal local preferences, the path with the higher local preference is chosen.



Note If a local preference attribute was received with the path or was set by a routing policy, then that value is used in this comparison. Otherwise, the default local preference value of 100 is used. The default value can be changed using the **bgp default local-preference** command.

5. If one of the paths is a redistributed path, which results from a **redistribute** or **network** command, then it is chosen. Otherwise, if one of the paths is a locally generated aggregate, which results from an **aggregate-address** command, it is chosen.



Note Step 1 through Step 4 implement the “Path Selection with BGP” of RFC 1268.

6. If the paths have unequal AS path lengths, the path with the shorter AS path is chosen. This step is skipped if **bgp bestpath as-path ignore** command is configured.



Note When calculating the length of the AS path, confederation segments are ignored, and AS sets count as 1.



Note eBGP specifies internal and external BGP multipath peers. eBGP allows simultaneous use of internal and external paths.

7. If the paths have different origins, the path with the lower origin is selected. Interior Gateway Protocol (IGP) is considered lower than EGP, which is considered lower than INCOMPLETE.
8. If appropriate, the MED of the paths is compared. If they are unequal, the path with the lower MED is chosen.

A number of configuration options exist that affect whether or not this step is performed. In general, the MED is compared if both paths were received from neighbors in the same AS; otherwise the MED comparison is skipped. However, this behavior is modified by certain configuration options, and there are also some corner cases to consider.

If the **bgp bestpath med always** command is configured, then the MED comparison is always performed, regardless of neighbor AS in the paths. Otherwise, MED comparison depends on the AS paths of the two paths being compared, as follows:

- If a path has no AS path or the AS path starts with an AS_SET, then the path is considered to be internal, and the MED is compared with other internal paths.
- If the AS path starts with an AS_SEQUENCE, then the neighbor AS is the first AS number in the sequence, and the MED is compared with other paths that have the same neighbor AS.

- If the AS path contains only confederation segments or starts with confederation segments followed by an AS_SET, then the MED is not compared with any other path unless the **bgp bestpath med confed** command is configured. In that case, the path is considered internal and the MED is compared with other internal paths.
- If the AS path starts with confederation segments followed by an AS_SEQUENCE, then the neighbor AS is the first AS number in the AS_SEQUENCE, and the MED is compared with other paths that have the same neighbor AS.



Note If no MED attribute was received with the path, then the MED is considered to be 0 unless the **bgp bestpath med missing-as-worst** command is configured. In that case, if no MED attribute was received, the MED is considered to be the highest possible value.

9. If one path is received from an external peer and the other is received from an internal (or confederation) peer, the path from the external peer is chosen.
10. If the paths have different IGP metrics to their next hops, the path with the lower IGP metric is chosen.
11. If the paths have unequal IP cost communities, the path with the lower IP cost community is selected as the best path.
12. If all path parameters in Step 1 through Step 10 are the same, then the router IDs are compared. If the path was received with an originator attribute, then that is used as the router ID to compare; otherwise, the router ID of the neighbor from which the path was received is used. If the paths have different router IDs, the path with the lower router ID is chosen.



Note Where the originator is used as the router ID, it is possible to have two paths with the same router ID. It is also possible to have two BGP sessions with the same peer router, and therefore receive two paths with the same router ID.

13. If the paths have different cluster lengths, the path with the shorter cluster length is selected. If a path was not received with a cluster list attribute, it is considered to have a cluster length of 0.
14. Finally, the path received from the neighbor with the lower IP address is chosen. Locally generated paths (for example, redistributed paths) are considered to have a neighbor IP address of 0.

Order of Comparisons

The second part of the BGP best-path algorithm implementation determines the order in which the paths should be compared. The order of comparison is determined as follows:

1. The paths are partitioned into groups such that within each group the MED can be compared among all paths. The same rules as in *Comparing Paths* section are used to determine whether MED can be compared between any two paths. Normally, this comparison results in one group for each neighbor AS. If the **bgp bestpath med always** command is configured, then there is just one group containing all the paths.
2. The best path in each group is determined. Determining the best path is achieved by iterating through all paths in the group and keeping track of the best one seen so far. Each path is compared with the best-so-far, and if it is better, it becomes the new best-so-far and is compared with the next path in the group.

3. A set of paths is formed containing the best path selected from each group in Step 2. The overall best path is selected from this set of paths, by iterating through them as in Step 2.

Best Path Change Suppression

The third part of the implementation is to determine whether the best-path change can be suppressed or not—whether the new best path should be used, or continue using the existing best path. The existing best path can continue to be used if the new one is identical to the point at which the best-path selection algorithm becomes arbitrary (if the router-id is the same). Continuing to use the existing best path can avoid churn in the network.



Note This suppression behavior does not comply with the IETF Networking Working Group draft-ietf-idr-bgp4-24.txt document, but is specified in the IETF Networking Working Group draft-ietf-idr-avoid-transition-00.txt document.

The suppression behavior can be turned off by configuring the **bgp bestpath compare-routerid** command. If this command is configured, the new best path is always preferred to the existing one.

Otherwise, the following steps are used to determine whether the best-path change can be suppressed:

1. If the existing best path is no longer valid, the change cannot be suppressed.
2. If either the existing or new best paths were received from internal (or confederation) peers or were locally generated (for example, by redistribution), then the change cannot be suppressed. That is, suppression is possible only if both paths were received from external peers.
3. If the paths were received from the same peer (the paths would have the same router-id), the change cannot be suppressed. The router ID is calculated using rules in *Comparing Pairs of Paths* section.
4. If the paths have different weights, local preferences, origins, or IGP metrics to their next hops, then the change cannot be suppressed. Note that all these values are calculated using the rules in *Comparing Pairs of Paths* section..
5. If the paths have different-length AS paths and the **bgp bestpath as-path ignore** command is not configured, then the change cannot be suppressed. Again, the AS path length is calculated using the rules in *Comparing Pairs of Paths* section.
6. If the MED of the paths can be compared and the MEDs are different, then the change cannot be suppressed. The decision as to whether the MEDs can be compared is exactly the same as the rules in *Comparing Pairs of Paths* section, as is the calculation of the MED value.
7. If all path parameters in Step 1 through Step 6 do not apply, the change can be suppressed.

Administrative Distance

An administrative distance is a rating of the trustworthiness of a routing information source. In general, the higher the value, the lower the trust rating.

Normally, a route can be learned through more than one protocol. Administrative distance is used to discriminate between routes learned from more than one protocol. The route with the lowest administrative distance is installed in the IP routing table. By default, BGP uses the administrative distances shown in *BGP Default Administrative Distances* section.

Table 18: BGP Default Administrative Distances

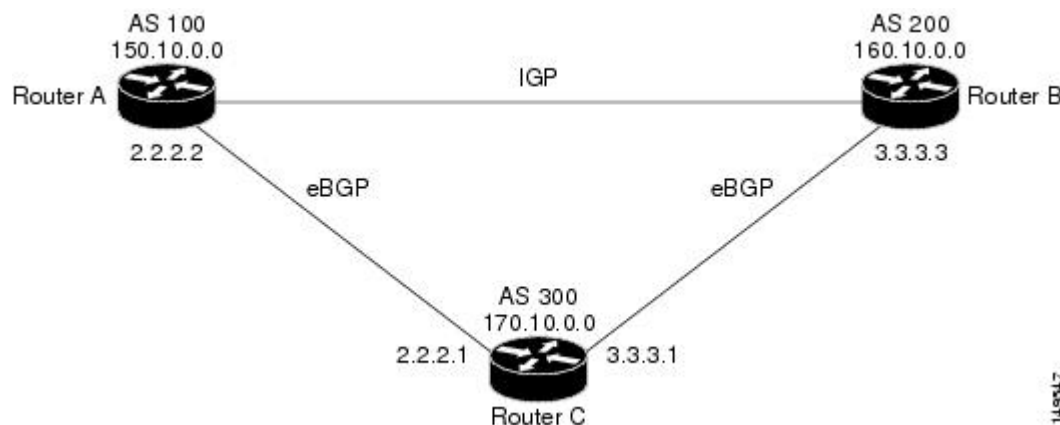
Distance	Default Value	Function
External	20	Applied to routes learned from eBGP.
Internal	200	Applied to routes learned from iBGP.
Local	200	Applied to routes originated by the router.



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

In most cases, when a route is learned through eBGP, it is installed in the IP routing table because of its distance (20). Sometimes, however, two ASs have an IGP-learned back-door route and an eBGP-learned route. Their policy might be to use the IGP-learned path as the preferred path and to use the eBGP-learned path when the IGP path is down.

Figure 7: Back Door Example



In *Back Door Example* section, Routers A and C and Routers B and C are running eBGP. Routers A and B are running an IGP (such as Routing Information Protocol [RIP], Interior Gateway Routing Protocol [IGRP], Enhanced IGRP, or Open Shortest Path First [OSPF]). The default distances for RIP, IGRP, Enhanced IGRP, and OSPF are 120, 100, 90, and 110, respectively. All these distances are higher than the default distance of eBGP, which is 20. Usually, the route with the lowest distance is preferred.

Router A receives updates about 160.10.0.0 from two routing protocols: eBGP and IGP. Because the default distance for eBGP is lower than the default distance of the IGP, Router A chooses the eBGP-learned route from Router C. If you want Router A to learn about 160.10.0.0 from Router B (IGP), establish a BGP back door. See .

In the following example, a network back-door is configured:

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

Router A treats the eBGP-learned route as local and installs it in the IP routing table with a distance of 200. The network is also learned through Enhanced IGRP (with a distance of 90), so the Enhanced IGRP route is successfully installed in the IP routing table and is used to forward traffic. If the Enhanced IGRP-learned route goes down, the eBGP-learned route is installed in the IP routing table and is used to forward traffic.

Although BGP treats network 160.10.0.0 as a local entry, it does not advertise network 160.10.0.0 as it normally would advertise a local entry.

Route Dampening

Route dampening is a BGP feature that minimizes the propagation of flapping routes across an internetwork. A route is considered to be flapping when it is repeatedly available, then unavailable, then available, then unavailable, and so on.

For example, consider a network with three BGP autonomous systems: autonomous system 1, autonomous system 2, and autonomous system 3. Suppose the route to network A in autonomous system 1 flaps (it becomes unavailable). Under circumstances without route dampening, the eBGP neighbor of autonomous system 1 to autonomous system 2 sends a withdraw message to autonomous system 2. The border router in autonomous system 2, in turn, propagates the withdrawal message to autonomous system 3. When the route to network A reappears, autonomous system 1 sends an advertisement message to autonomous system 2, which sends it to autonomous system 3. If the route to network A repeatedly becomes unavailable, then available, many withdrawal and advertisement messages are sent. Route flapping is a problem in an internetwork connected to the Internet, because a route flap in the Internet backbone usually involves many routes.

Minimize Flapping

The route dampening feature minimizes the flapping problem as follows. Suppose again that the route to network A flaps. The router in autonomous system 2 (in which route dampening is enabled) assigns network A a penalty of 1000 and moves it to history state. The router in autonomous system 2 continues to advertise the status of the route to neighbors. The penalties are cumulative. When the route flaps so often that the penalty exceeds a configurable suppression limit, the router stops advertising the route to network A, regardless of how many times it flaps. Thus, the route is dampened.

The penalty placed on network A is decayed until the reuse limit is reached, upon which the route is once again advertised. At half of the reuse limit, the dampening information for the route to network A is removed.



Note No penalty is applied to a BGP peer reset when route dampening is enabled, even though the reset withdraws the route.

BGP Routing Domain Confederation

One way to reduce the iBGP mesh is to divide an autonomous system into multiple sub-autonomous systems and group them into a single confederation. To the outside world, the confederation looks like a single autonomous system. Each autonomous system is fully meshed within itself and has a few connections to other autonomous systems in the same confederation. Although the peers in different autonomous systems have eBGP sessions, they exchange routing information as if they were iBGP peers. Specifically, the next hop, MED, and local preference information is preserved. This feature allows you to retain a single IGP for all of the autonomous systems.

BGP Optimal Route Reflector

BGP-ORR (optimal route reflector) enables virtual route reflector (vRR) to calculate the best path from a route reflector (RR) client's point of view.

BGP ORR calculates the best path by:

1. Running SPF multiple times in the context of its RR clients or RR clusters (set of RR clients)
2. Saving the result of different SPF runs in separate databases
3. Using these databases to manipulate BGP best path decision and thereby allowing BGP to use and announce best path that is optimal from the client's point of view

**Note**

- Enabling the BGP ORR feature increases the memory footprint of BGP and RIB. With increased number of vRR configured in the network, ORR adversely impacts convergence for BGP.
- The BGP ORR configuration allows you to set up to 256 ORR groups. However, this does not ensure that the route reflector can always handle all 256 groups, as its capacity depends on the scale and activation of other BGP or IGP features. As a best practice, we recommend you to configure no more than 32 BGP ORR groups and limit each group to 60 BGP clients in classical BGP Route Reflector setups.

In an autonomous system, a BGP route reflector acts as a focal point and advertises routes to its peers (RR clients) along with the RR's computed best path. Since the best path advertised by the RR is computed from the RR's point of view, the RR's placement becomes an important deployment consideration.

With network function virtualization (NFV) becoming a dominant technology, service providers (SPs) are hosting virtual RR functionality in a cloud using servers. A vRR can run on a control plane device and can be placed anywhere in the topology or in a SP data center. Cisco IOS XRv 9000 Router can be implemented as vRR over a NFV platform in a SP data center. vRR allows SPs to scale memory and CPU usage of RR deployments significantly. Moving a RR out of its optimal placement requires vRRs to implement ORR functionality that calculates the best path from a RR client's point of view.

BGP ORR offers these benefits:

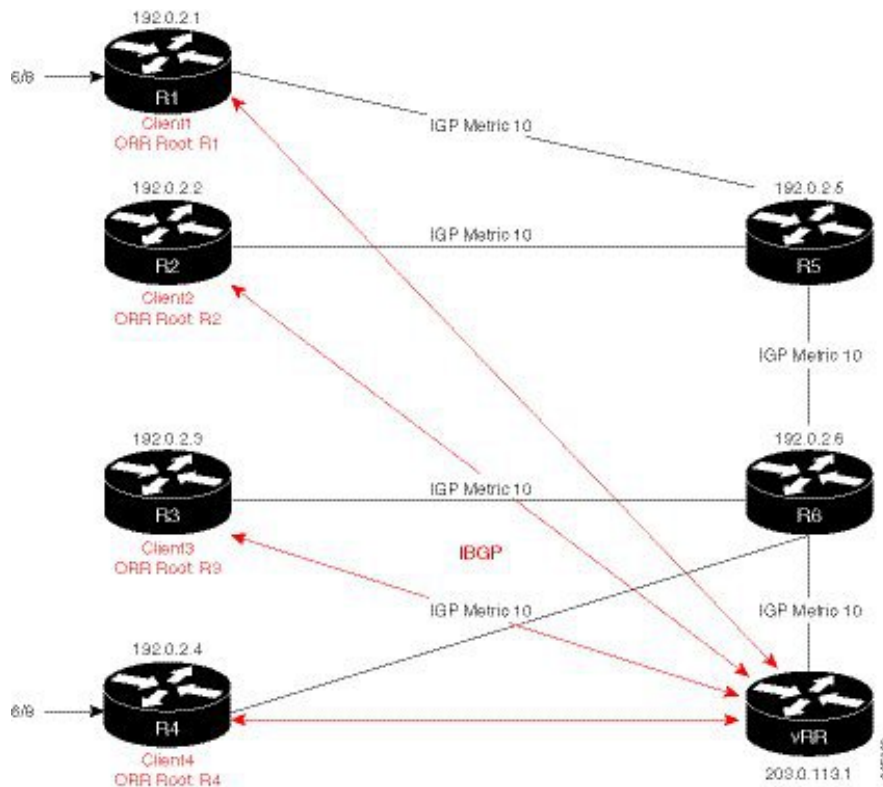
- Calculates the bestpath from the point of view of a RR client.
- Enables vRR to be placed anywhere in the topology or in a SP data center.
- Allows SPs to scale memory and CPU usage of RR deployments.

Use Case

Consider a BGP Route Reflector topology where:

- Router R1, R2, R3, R4, R5 and R6 are route reflector clients
- Router R1 and R4 advertise 6/8 prefix to vRR

Figure 8: BGP-ORR Topology



vRR receives prefix 6/8 from R1 and R4. Without BGP ORR configured in the network, the vRR selects R4 as the closest exit point for RR clients R2, R3, R5, and R6, and reflects the 6/8 prefix learned from R4 to these RR clients R2, R3, R5, and R6. From the topology, it is evident that for R2 the best path is R1 and not R4. This is because the vRR calculates best path from the RR's point of view.

When the BGP ORR is configured in the network, the vRR calculates the shortest exit point in the network from R2's point of view (ORR Root: R2) and determines that R1 is the closest exit point to R2. vRR then reflects the 6/8 prefix learned from R1 to R2.

Configuring BGP ORR includes:

- enabling ORR on the RR for the client whose shortest exit point is to be determined
- applying the ORR configuration to the neighbor

Enabling ORR on vRR for R2 (RR client)

For example to determine shortest exit point for R2; configure ORR on vRR with an IP address of R2 that is 192.0.2.2. Use 6500 as AS number and g1 as orr (root) policy name:

```
router bgp 6500
 address-family ipv4 unicast
  optimal-route-reflection g1 192.0.2.2
commit
```

Applying the ORR configuration to the neighbor

Next, apply the ORR policy to BGP neighbor R2 (this enables RR to advertise best path calculated using the root IP address, 192.0.2.2, configured in orr (root) policy g1 to R2):

```
router bgp 6500
 neighbor 192.0.2.2
   address-family ipv4 unicast
     optimal-route-reflection g1
commit
```

Configuring MPLS Traffic-Engineering on Root Router

The root routers advertise the Multi Protocol Label Switching (MPLS) TE router-ID that matches with the configured root address on the RR. So, you must configure the root router with a minimal MPLS TE configuration to advertise this MPLS TE router-ID. The minimal set of commands that you need to configure depends on the operating system of the root router.

The following is a sample configuration on the root router:

```
router isis 100

is-type level-2-only

net 49.0001.0000.0000.0001.00

distribute link-state

    metric-style wide

    mpls traffic-eng level-2-only

    mpls traffic-eng router-id Loopback0

!

mpls traffic-eng
```

Verification

To verify whether R2 received the best exit, execute the **show bgp <prefix>** command (from R2) in EXEC mode. In the above example, R1 and R4 advertise the 6/8 prefix; run the **show bgp 6.0.0.0/8** command:

```
R2# show bgp 6.0.0.0/8
Tue Apr  5 20:21:58.509 UTC
BGP routing table entry for 6.0.0.0/8
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 203.0.113.1 (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
```

The above show output states that the best path for R2 is through R1, whose IP address is 192.0.2.1 and the metric of the path is 20.

Execute the **show bgp** command from the vRR to determine the best path calculated for R2 by ORR. R2 has its own update-group because it has a different best path (or different policy configured) than those of other peers:

```
VRR#show bgp 6.0.0.0/8
Thu Apr 28 13:36:42.744 UTC
BGP routing table entry for 6.0.0.0/8
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
```



Note Path #1 is advertised to update-group 0.1. R2 is in update-group 0.1.

Execute the **show bgp** command for update-group 0.1 verify whether R2 is in update-group 0.1.

```
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

```

For further verification, check the contents of the table created on vRR as a result of configuring the g1 policy. From R2's point of view, the cost of reaching R1 is 20 and the cost of reaching R4 is 30. Therefore, the closest and best exit for R2 is through R1:

```

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

```

VPN Route Limit on Route Reflectors

With the VPN Route Limit feature, you can configure Route Reflectors (RRs) to retain only a certain number of unique network entries for each VPN. This limit is defined by a set of Route Targets (RTs) associated with the VPN. You can set the maximum number of routes that an RR accepts from a particular VPN, ensuring that the RR's resources are used efficiently.

Per-VPN Configuration

The maximum route limit is configurable on a per-VPN basis, allowing for different VPNs to have unique limits according to their individual requirements.

Selective Route Dropping

When 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 has exceeded its limit, and it does not affect other VPNs or active BGP sessions.

Route Count Mechanism

BGP maintains a route count for each unique set of RTs. This count reflects the number of prefixes that have at least one path that is tagged with the corresponding RT-set. This count is incremented by one for each prefix, regardless of the number of paths sharing the same RT-set. In scenarios involving multiple RRs, a path accepted by one RR results in the acceptance of identical paths from other RRs, promoting consistency across the network.

Even if the RT-set limit is specified in the neighbor's inbound route-policy, the RT-set is global to the VPN address-family. The route count is not per neighbor.

Inbound RPL Policy and Route Acceptance

When BGP receives a path from a neighbor, it is evaluated against the inbound Route Policy Language (RPL) policy. The inbound RPL sets an RT-set limit for the path, and the BGP checks the current count for the RT-Set. If the count is below the limit, or if the prefix already has a path with the same RT-set, the path is accepted. Otherwise, the path is dropped.

Guidelines and Recommendations

When the VPN route limit is reached, the routes from a neighbor may vary if the neighbor experiences a flap. This is because the dropping of routes is entirely dependent on the order in which the routes are received.

To protect the Route Reflector (RR) and Provider Edge (PE) devices, we recommend you to configure the VPN route limit to be 20 percent higher than the expected scale.

Limitations for VPN Route Limit

These limitations apply for the VPN route limit feature.

- When the VPN route limit feature is enabled, the active and standby RRs may have different prefixes and paths. This happens because the active and standby RRs receive updates independently, and the RRs do not guarantee the sequence of prefixes. So, Non-Stop Routing (NSR) is not supported with the VPN Route Limit feature.
- If the policy is modified to reduce the VPN route limit from a higher value to a lower value (for example from 200 to 50), the updated limit is enforced exclusively on single path networks. Networks with multiple paths are not subjected to this new route limit, and 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, then the routing behavior is nondeterministic.

Configure VPN Route Limit

Procedure

Step 1 Enable BGP routing and configure a route policy.

Example:

```
Router(config)# router bgp 100
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 2 Configure the VPN route limit using the **set rt-set route-limit limit-value** command in route-policy configuration mode.

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 3 To verify your configuration, use the **show bgp vpnv4 unicast rt-set** or **show bgp vpnv4 unicast path rt-set** command.

Example:

```
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

Example:

```
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
```

BGP Functional Overview

Table 19: Feature History Table

Feature Name	Release Information	
BGP Functional Overview	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) Enhances routing efficiency by enabling Border Gateway Protocol (BGP) to exchange routing information between autonomous systems. Supports route filtering, route aggregation, and policy-based routing, allowing for precise control over routing decisions. The feature is essential for large-scale network environments requiring scalability and stability. It includes improvements in route reflection and confederation to optimize routing paths and decrease the burden on network resources. Additionally, it facilitates seamless integration with existing network infrastructure, ensuring reliable and efficient data transfer across diverse networks. *This functionality is now supported on Cisco 8712-MOD-M routers.

BGP uses TCP as its transport protocol. Two BGP routers form a TCP connection between one another (peer routers) and exchange messages to open and confirm the connection parameters.

BGP routers exchange network reachability information. This information is mainly an indication of the full paths (BGP autonomous system numbers) that a route should take to reach the destination network. This information helps construct a graph that shows which autonomous systems are loop free and where routing policies can be applied to enforce restrictions on routing behavior.

Any two routers forming a TCP connection to exchange BGP routing information are called peers or neighbors. BGP peers initially exchange their full BGP routing tables. After this exchange, incremental updates are sent as the routing table changes. BGP keeps a version number of the BGP table, which is the same for all of its BGP peers. The version number changes whenever BGP updates the table due to routing information changes. Keepalive packets are sent to ensure that the connection is alive between the BGP peers and notification packets are sent in response to error or special conditions.



Note ASN change for BGP process is not currently supported via **commit replace**.

RPL - if prefix is-best-path/is-best-multipath

Border Gateway Protocol (BGP) routers receive multiple paths to the same destination. As a standard, by default the BGP best path algorithm decides the best path to install in IP routing table. This is used for traffic forwarding.

BGP assigns the first valid path as the current best path. It then compares the best path with the next in the list. This process continues, until BGP reaches the end of the list of valid paths. This contains all rules used to determine the best path. When there are multiple paths for a given address prefix, BGP:

- Selects one of the paths as the best path as per the best-path selection rules.

- Installs the best path in its forwarding table. Each BGP speaker advertises only the best-path to its peers.



Note The advertisement rule of sending only the best path does not convey the full routing state of a destination, present on a BGP speaker to its peers.

After the BGP speaker receives a path from one of its peers; the path is used by the peer for forwarding packets. All other peers receive the same path from this peer. This leads to a consistent routing in a BGP network. To improve the link bandwidth utilization, most BGP implementations choose additional paths satisfy certain conditions, as multi-path, and install them in the forwarding table. Incoming packets for such are load-balanced across the best-path and the multi-path(s). You can install the paths in the forwarding table that are not advertised to the peers. The RR route reflector finds out the best-path and multi-path. This way the route reflector uses different communities for best-path and multi-path. This feature allows BGP to signal the local decision done by RR or Border Router. With this new feature, selected by RR using community-string (if is-best-path then community 100:100). The controller checks which best path is sent to all R's. Border Gateway Protocol routers receive multiple paths to the same destination. While carrying out best path computation there will be one best path, sometimes equal and few non-equal paths. Thus, the requirement for a best-path and is-equal-best-path.

The BGP best path algorithm decides the best path in the IP routing table and used for forwarding traffic. This enhancement within the RPL allows creating policy to take decisions. Adding community-string for local selection of best path. With introduction of BGP Additional Path (Add Path), BGP now signals more than the best Path. BGP can signal the best path and the entire path equivalent to the best path. This is in accordance to the BGP multi-path rules and all backup paths.

Remotely Triggered Blackhole Filtering with RPL Next-hop Discard Configuration

Remotely triggered black hole (RTBH) filtering is a technique that provides the ability to drop undesirable traffic before it enters a protected network. RTBH filtering provides a method for quickly dropping undesirable traffic at the edge of the network, based on either source addresses or destination addresses by forwarding it to a null0 interface. RTBH filtering based on a destination address is commonly known as Destination-based RTBH filtering. Whereas, RTBH filtering based on a source address is known as Source-based RTBH filtering.

RTBH filtering is one of the many techniques in the security toolkit that can be used together to enhance network security in the following ways:

- Effectively mitigate DDoS and worm attacks
- Quarantine all traffic destined for the target under attack
- Enforce blocklist filtering

Configure Destination-based RTBH Filtering

RTBH is implemented by defining a route policy (RPL) to discard undesirable traffic at next-hop using **set next-hop discard** command.

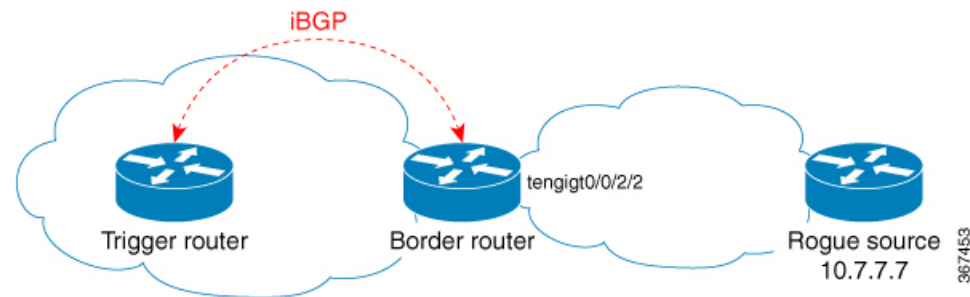
RTBH filtering sets the next-hop of the victim's prefix to the null interface. The traffic destined to the victim is dropped at the ingress.

The **set next-hop discard** configuration is used in the neighbor inbound policy. When this config is applied to a path, though the primary next-hop is associated with the actual path but the RIB is updated with next-hop set to Null0. Even if the primary received next-hop is unreachable, the RTBH path is considered reachable and will be a candidate in the bestpath selection process. The RTBH path is readvertised to other peers with either the received next-hop or nexthop-self based on normal BGP advertisement rules.

A typical deployment scenario for RTBH filtering would require running internal Border Gateway Protocol (iBGP) at the access and aggregation points and configuring a separate device in the network operations center (NOC) to act as a trigger. The triggering device sends iBGP updates to the edge, that cause undesirable traffic to be forwarded to a null0 interface and dropped.

Consider below topology, where a rogue router is sending traffic to a border router.

Figure 9: Topology to Implement RTBH Filtering



Configurations applied on the Trigger Router

Configure a static route redistribution policy that sets a community on static routes marked with a special tag, and apply it in BGP:

```
route-policy RTBH-trigger
  if tag is 777 then
    set community (1234:4321, no-export) additive
    pass
  else
    pass
  endif
end-policy

router bgp 65001
  address-family ipv4 unicast
    redistribute static route-policy RTBH-trigger
  !
  neighbor 192.168.102.1
    remote-as 65001
    address-family ipv4 unicast
    route-policy bgp_all in
    route-policy bgp_all out
```

Configure a static route with the special tag for the source prefix that has to be block-holed:

```
router static
  address-family ipv4 unicast
  10.7.7.7/32 Null0 tag 777
```

Configurations applied on the Border Router

Configure a route policy that matches the community set on the trigger router and configure set next-hop discard:

```
route-policy RTBH
  if community matches-any (1234:4321) then
    set next-hop discard
  else
    pass
  endif
end-policy
```

Apply the route policy on the iBGP peers:

```
router bgp 65001
  address-family ipv4 unicast
  !
  neighbor 192.168.102.2
  remote-as 65001
  address-family ipv4 unicast
    route-policy RTBH in
    route-policy bgp_all out
```

Default Address Family for show Commands

Most of the **show** commands provide address family (AFI) and subaddress family (SAFI) arguments (see RFC 1700 and RFC 2858 for information on AFI and SAFI). The Cisco IOS XR software parser provides the ability to set the afi and safi so that it is not necessary to specify them while running a **show** command. The parser commands are:

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

The parser automatically sets the default afi value to **ipv4** and default safi value to **unicast**. It is necessary to use only the parser commands to change the default afi value from **ipv4** or default safi value from **unicast**. Any **afi** or **safi** keyword specified in a **show** command overrides the values set using the parser commands. Use the following **show default-afi-safi-vrf** command to check the currently set value of the afi and safi.

TCP Maximum Segment Size

Maximum Segment Size (MSS) is the largest amount of data that a computer or a communication device can receive in a single, unfragmented TCP segment. All TCP sessions are bounded by a limit on the number of bytes that can be transported in a single packet; this limit is MSS. TCP breaks up packets into chunks in a transmit queue before passing packets down to the IP layer.

The TCP MSS value is dependent on the maximum transmission unit (MTU) of an interface, which is the maximum length of data that can be transmitted by a protocol at one instance. The maximum TCP packet length is determined by both the MTU of the outbound interface on the source device and the MSS announced by the destination device during the TCP setup process. The closer the MSS is to the MTU, the more efficient is the transfer of BGP messages. Each direction of data flow can use a different MSS value.

Per Neighbor TCP MSS

The per neighbor TCP MSS feature allows you to create unique TCP MSS profiles for each neighbor. Per neighbor TCP MSS is supported in two modes: neighbor group and session group. Before, TCP MSS configuration was available only at the global level in the BGP configuration.

The per neighbor TCP MSS feature allows you to:

- Enable per neighbor TCP MSS configuration.
- Disable TCP MSS for a particular neighbor in the neighbor group or session group using the **inheritance-disable** command.
- Unconfigure TCP MSS value. On unconfiguration, TCP MSS value in the protocol control block (PCB) is set to the default value.



Note The default TCP MSS value is 536 (in octets) or 1460 (in bytes). The MSS default of 1460 means that TCP segments the data in the transmit queue into 1460-byte chunks before passing the packets to the IP layer.

To configure per neighbor TCP MSS, use the **tcp mss** command under per neighbor, neighbor group or session group configuration.

For detailed configuration steps, see the *Configuring Per Neighbor TCP MSS* section.

For detailed steps to disable per neighbor TCP MSS, see the *Disabling Per Neighbor TCP MSS* section.

BGP Keychains

BGP keychains enable keychain authentication between two BGP peers. The BGP endpoints must both comply with draft-bonica-tcp-auth-05.txt and a keychain on one endpoint and a password on the other endpoint does not work.

BGP is able to use the keychain to implement hitless key rollover for authentication. Key rollover specification is time based, and in the event of clock skew between the peers, the rollover process is impacted. The configurable tolerance specification allows for the accept window to be extended (before and after) by that margin. This accept window facilitates a hitless key rollover for applications (for example, routing and management protocols).

The key rollover does not impact the BGP session, unless there is a keychain configuration mismatch at the endpoints resulting in no common keys for the session traffic (send or accept).

BGP Nonstop Routing

The Border Gateway Protocol (BGP) Nonstop Routing (NSR) with Stateful Switchover (SSO) feature enables all bgp peerings to maintain the BGP state and ensure continuous packet forwarding during events that could interrupt service. Under NSR, events that might potentially interrupt service are not visible to peer routers. Protocol sessions are not interrupted and routing states are maintained across process restarts and switchovers.

BGP NSR provides nonstop routing during the following events:

- Route processor switchover

- Process crash or process failure of BGP or TCP



Note BGP NSR is enabled by default. Use the **nsr disable** command to turn off BGP NSR. The **no nsr disable** command can also be used to turn BGP NSR back on if it has been disabled.

In case of process crash or process failure, NSR will be maintained only if **nsr process-failures switchover** command is configured. In the event of process failures of active instances, the **nsr process-failures switchover** configures failover as a recovery action and switches over to a standby route processor (RP) or a standby distributed route processor (DRP) thereby maintaining NSR. An example of the configuration command is `RP/0/RSP0/CPU0:router(config) # nsr process-failures switchover`

The **nsr process-failures switchover** command maintains both the NSR and BGP sessions in the event of a BGP or TCP process crash. Without this configuration, BGP neighbor sessions flap in case of a BGP or TCP process crash. This configuration does not help if the BGP or TCP process is restarted in which case the BGP neighbors are expected to flap.

When the *l2vpn_mgr* process is restarted, the NSR client (te-control) flaps between the **Ready** and **Not Ready** state. This is the expected behavior and there is no traffic loss.

During route processor switchover and In-Service System Upgrade (ISSU), NSR is achieved by stateful switchover (SSO) of both TCP and BGP.

NSR does not force any software upgrades on other routers in the network, and peer routers are not required to support NSR.

When a route processor switchover occurs due to a fault, the TCP connections and the BGP sessions are migrated transparently to the standby route processor, and the standby route processor becomes active. The existing protocol state is maintained on the standby route processor when it becomes active, and the protocol state does not need to be refreshed by peers.

Events such as soft reconfiguration and policy modifications can trigger the BGP internal state to change. To ensure state consistency between active and standby BGP processes during such events, the concept of post-it is introduced that act as synchronization points.

BGP NSR provides the following features:

- NSR-related alarms and notifications
- Configured and operational NSR states are tracked separately
- NSR statistics collection
- NSR statistics display using **show** commands
- XML schema support
- Auditing mechanisms to verify state synchronization between active and standby instances
- CLI commands to enable and disable NSR
- Support for 5000 NSR sessions

BGP Best-External Path

The best-external path functionality supports advertisement of the best-external path to the iBGP and Route Reflector peers when a locally selected bestpath is from an internal peer. BGP selects one best path and one backup path to every destination. By default, selects one best path. Additionally, BGP selects another bestpath from among the remaining external paths for a prefix. Only a single path is chosen as the best-external path and is sent to other PEs as the backup path. BGP calculates the best-external path only when the best path is an iBGP path. If the best path is an eBGP path, then best-external path calculation is not required.

The procedure to determine the best-external path is as follows:

1. Determine the best path from the entire set of paths available for a prefix.
2. Eliminate the current best path.
3. Eliminate all the internal paths for the prefix.
4. From the remaining paths, eliminate all the paths that have the same next hop as that of the current best path.
5. Rerun the best path algorithm on the remaining set of paths to determine the best-external path.

BGP considers the external and confederations BGP paths for a prefix to calculate the best-external path. BGP advertises the best path and the best-external path as follows:

- On the primary PE—advertises the best path for a prefix to both its internal and external peers
- On the backup PE—advertises the best path selected for a prefix to the external peers and advertises the best-external path selected for that prefix to the internal peers

BGP Prefix Independent Convergence

Table 20: Feature History Table

Feature Name	Release	Description
BGP Prefix Independent Convergence	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700)(select variants only*).</p> <p>This feature ensures rapid network recovery by activating a pre-established backup path when the primary path fails, minimizing downtime in large-scale network deployments.</p> <p>*This feature is now supported on Cisco 8712-MOD-M Routers.</p>

BGP Prefix Independent Convergence (PIC) feature enables the activation of a backup path in the event of the primary path failure.

Networks use Fast reroute (FRR) to calculate the next best path (backup path) and store it in BGP and IP Routing Information Bases (RIBs). The RIBs share the backup path information with the Forwarding

Information Base (FIB). BGP PIC feature uses the backup path information in the FIB to quickly switch to this path during network failure, provided the line cards are enabled for PIC.

Drawbacks of Using Prefix-Dependent Convergence

In a standard BGP network, a BGP router advertises only its best path to a destination prefix. Hence, in an autonomous system, routers running BGP are not aware of all the possible paths to a destination prefix. In the event of a link or network failure that causes the best path to fail, the following process takes place:

1. The affected BGP router advertising the failed best path, announces a withdrawal of the path.
2. The BGP routers receiving the best path withdrawal from the affected BGP router, withdraw their own best paths, and recalculate their best paths to the destination prefix.
3. The BGP routers advertise their recalculated best paths to all neighboring routers.
4. Each BGP router that receives a new best path from its neighboring BGP router, again evaluates its own best path, and possibly withdraws and recalculates its best path.
5. The BGP routers that recalculate their best paths, again advertise the new paths in the network.

Because this process repeats until all the BGP routers have the best path to the destination prefix, convergence of the network takes a lot of time. This form of convergence is known as prefix-dependent convergence. If route reflectors are configured in the network, then convergence takes even longer.

Benefits of Using Prefix-Independent Convergence

When prefix-independent convergence is configured in a BGP network, all BGP routers advertise their best external paths to a destination prefix. This indicates that all BGP routers are aware of multiple best external paths to a destination prefix.

Each BGP router selects a backup path from the available best external paths, and downloads it to its FIB. Hence, the FIB on each BGP router contains a best path and a best external path to a destination prefix. In the event of a link or network failure that causes the best path to fail, the FIB on the affected BGP router can switch all its routes using the failed path to the best external path, in a single operation. Because this form of convergence takes minimal time, it is preferred in large scale network deployments.

Using Prefix-Independent Convergence with Route Reflectors

For traffic from the customer edge router to a remote provider edge router, the BGP `local-pref` attribute is used to select the primary path (from a primary PE) and the backup path (from the backup PE). Even though the remote provider edge router receives the backup (best external) path from the backup PE, when the backup PE receives the iBGP best path from the primary PE, it withdraws the backup path from the core network. Hence, the primary and backup (best external) paths must be pre-programmed in the network for PIC to work.

When the primary path fails, the delay in convergence is because of the following process that takes place:

1. The primary PE sends a request to the provider core network for withdrawing the primary path.
2. The backup PE advertises the backup (best external) path as the new primary (best) path.
3. The remote PE recalculates its primary paths on receiving the withdrawal request from the primary PE, and the new primary path from the backup PE.
4. Traffic resumes in the network after all prefixes in the FIB are updated with the new primary path.

Hence, convergence is slow because it depends on prefixes advertised by the PE routers.

By introducing prefix-independent convergence, the following changes take place:

- Primary and backup paths are pre-programmed in the RIB and FIB.
- All provider edge routers receive the backup path from the FIB.
- In the event of primary path failure, the FIB modifies LDIs to include the backup path and instantly divert traffic along this route.



Note To use BGP PIC feature with route reflectors, the provider edge routers must be configured with unique route distinguishers (RDs) within the context of a VRF. Else, the paths from different PEs are considered to be belonging to the same network, and the route reflector cannot accurately calculate the best backup path.

Backup Path Selection Process

Use the following procedure to identify the best backup path to be programmed in the RIB and FIB.

1. Use the best path algorithm to identify the best path from the available set of paths for a prefix.
2. Eliminate the best path.
3. Eliminate all paths that have the same next hop as the best path.
4. Rerun the best path algorithm on the remaining set of paths to identify the best backup path.

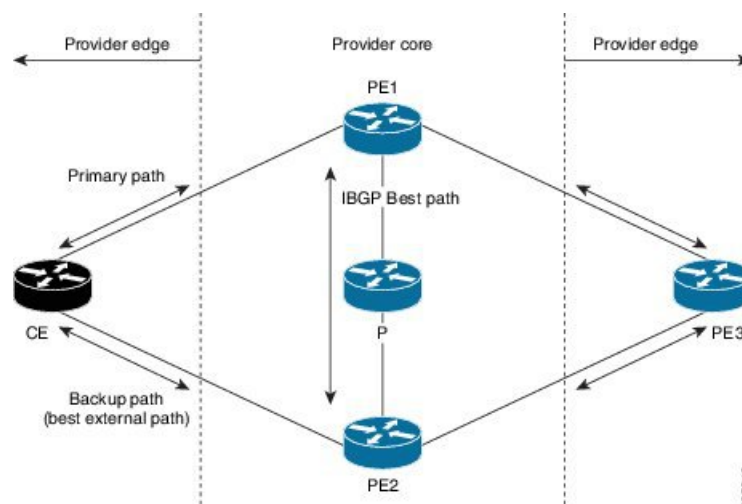
Configure BGP PIC in Provider Edge Networks

This section describes the procedure to configure BGP PIC for provider edge networks.

Topology

Consider the topology shown in the following illustration.

Figure 10: Prefix Independent Convergence in Provider Edge Networks



For traffic from the customer edge router CE to the provider edge router PE3, the BGP `local-pref` attribute is used to select CE-PE1-PE3 as the primary path, and CE-PE2-PE3 as the backup path. PE1-P-PE2 is the best internal path for the provider core network.

Before you Begin

Before you can configure the BGP PIC feature, ensure that you have configured the following:

1. The loopback and network interfaces as per the topology.
2. The VRFs for the provider core network.

Configuration

Use the configuration in this section to configure BGP PIC feature for the illustrated topology.

Router PE1

For traffic from Router CE to Router PE3, the eBGP path from Router CE is stored as the primary path on Router PE1.

Configure Router PE1 to install the backup (best external) path advertised by Router PE2, and the period for which the local label must be retained on convergence, as shown.

```
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
```

Router PE2

Configure Router PE2 to install and advertise the backup CE-PE2 path as the best external path.

```
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
```

Router PE3

The iBGP path from Router PE1 (CE-PE1) is stored as the primary path on Router PE3. Configure the iBGP backup path CE-PE2 as shown.

```
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
```

Verify BGP PIC

Run the following commands on Router PE3 to verify the BGP PIC feature in operation.

1. Verify the presence of the backup path in the FIB.

```
Router# show cef 1.1.1.1/32 detail
Fri Oct 10 10:24:33.079 UTC
1.1.1.1/32, 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 12.24.0.1, recursive
via 12.24.0.1, 3 dependencies, recursive
next hop 12.24.0.1 via 12.24.0.1/32
via 12.24.0.2, 3 dependencies, recursive, backup
next hop 12.24.0.2 via 12.24.0.2/32
Load distribution: 0 (refcount 1)
Hash OK Interface Address
0 Y MgmtEth0/RP0/CPU0/0 12.24.0.1

```

2. Verify the presence of the backup (best external) path for BGP.

```

Router# show bgp vrf foo 206.1.1.1/32
BGP routing table entry for 206.1.1.1/32
Versions:
Process bRIB/RIB SendTblVer
Speaker 6 6
Local Label: 3
Paths: (1 available, best #1)
Advertised to peers (in unique update groups):
100.100.100.1
Path #1: Received by speaker 0
1.1.1.1 from 1.1.1.1 (200.200.200.1)
Origin incomplete, metric 0, localpref 100, weight 32768, valid,
internal, best
2.2.2.2 from 2.2.2.2 (100.100.100.1)
Origin incomplete, metric 0, localpref 100, weight 32768, valid,
external, backup, best-external

```

Configure BGP PIC between Autonomous Systems

This section describes the procedure to configure BGP PIC between autonomous systems. .

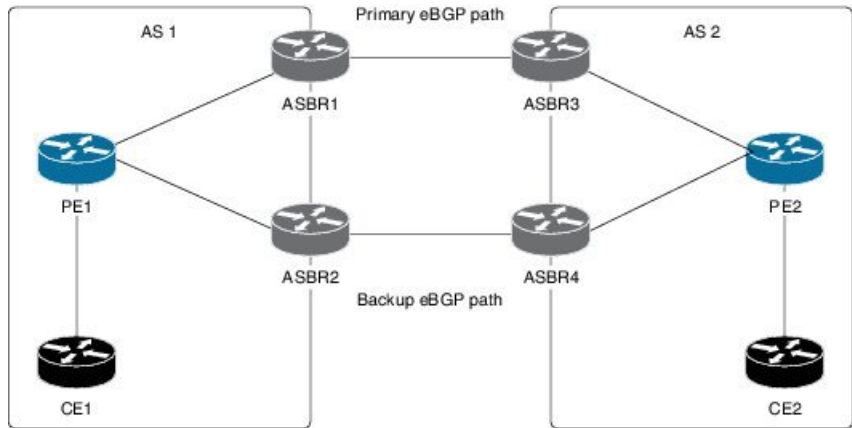


Note BGP PIC is supported only for Option A and Option B scenarios. The following section describes a sample configuration for Option B.

Topology

For example, consider the topology shown in the following illustration.

Figure 11: Prefix-Independent Convergence between Autonomous Systems



For traffic from Router PE1 to Router PE2, ASBR1 is the primary router and ASBR2 is the backup router. The ASBR1-ASBR3 eBGP path is the primary path. The ASBR2-ASBR4 eBGP path is the backup path. For traffic from Router PE2 to Router PE1, ASBR3 is the primary router and ASBR4 is the backup router. The ASBR3-ASBR1 eBGP path is the primary path and the ASBR4-ASBR2 eBGP path is the backup path.

Before you Begin

Before you can configure the BGP PIC feature, ensure that you have configured the loopback and network interfaces as per the illustrated topology.

Configuration

Use the configuration in this section to configure BGP PIC feature for the illustrated topology.

Router ASBR1

Configure Router ASBR1 to install the backup (best external) path advertised by Router ASBR2, and the period for which the local label must be retained on convergence, as shown.

```
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
```

The provided configuration is for traffic from Router PE1 to Router PE2. Similarly, configure Router ASBR3 for traffic from Router PE2 to Router PE1.

Router ASBR2

Configure Router ASBR2 to install and advertise the ASBR2-ASBR4 backup (best external) path, as shown.

```
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
```

The provided configuration is for traffic from Router PE1 to Router PE2. Similarly, configure Router ASBR4 for traffic from Router PE2 to Router PE1.

Verify BGP PIC

Run the following commands on Router PE2 (for traffic from Router PE1 to Router PE2) or on Router PE1 (for traffic from Router PE2 to Router PE1) to verify the BGP PIC feature in operation.

1. Verify the presence of the backup path in the FIB.

```
Router# show cef 1.1.1.1/32 detail

Fri Oct 10 10:24:33.079 UTC
1.1.1.1/32, 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 12.24.0.1, recursive
via 12.24.0.1, 3 dependencies, recursive
next hop 12.24.0.1 via 12.24.0.1/32
via 12.24.0.2, 3 dependencies, recursive, backup
next hop 12.24.0.2 via 12.24.0.2/32
Load distribution: 0 (refcount 1)
Hash OK Interface Address
0 Y MgmtEth0/RP0/CPU0/0 12.24.0.1
```

2. Verify the presence of the backup (best external) path for BGP.

```
Router# show bgp vrf foo 206.1.1.1/32

BGP routing table entry for 206.1.1.1/32
Versions:
Process bRIB/RIB SendTblVer
Speaker 6 6
Local Label: 3
Paths: (1 available, best #1)
Advertised to peers (in unique update groups):
100.100.100.1
Path #1: Received by speaker 0
1.1.1.1 from 1.1.1.1 (200.200.200.1)
Origin incomplete, metric 0, localpref 100, weight 32768, valid,
internal, best
2.2.2.2 from 2.2.2.2 (100.100.100.1)
Origin incomplete, metric 0, localpref 100, weight 32768, valid,
external, backup, best-external
```

Command Line Interface (CLI) Consistency for BGP Commands

The Border Gateway Protocol (BGP) commands use **disable** keyword to disable a feature. The keyword **inheritance-disable** disables the inheritance of the feature properties from the parent level.

BGP Additional Paths

Table 21: Feature History Table

Feature Name	Release Information	Feature Description

Additional path control per neighbor	Release 7.3.15	<p>This feature allows flexibility and granular control of the advertisement of additional paths based on the neighbor outbound policy configuration.</p> <p>This is done by allowing configuration of combinations of different path selection procedures unlike singular path selection, and extending neighbor outbound policy to have finer control of the path types to be advertised.</p> <p>This feature enables operational efficiency to manage additional paths and reduce scale of the paths in a typical clustered network architecture.</p> <p>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.</p>
--------------------------------------	----------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

The Border Gateway Protocol (BGP) Additional Paths feature modifies the BGP protocol machinery for a BGP speaker to be able to send multiple paths for a prefix. This gives 'path diversity' in the network. The add path enables BGP prefix independent convergence (PIC) at the edge routers.

BGP add path enables add path advertisement in an iBGP network and advertises the following types of paths for a prefix:

- Backup paths—to enable fast convergence and connectivity restoration.
- Group-best paths—to resolve route oscillation.
- All paths—to emulate an iBGP full-mesh.

iBGP Multipath Load Sharing

When a Border Gateway Protocol (BGP) speaking router that has no local policy configured, receives multiple network layer reachability information (NLRI) from the internal BGP (iBGP) for the same destination, the router will choose one iBGP path as the best path. The best path is then installed in the IP routing table of the router. The iBGP Multipath Load Sharing feature enables the BGP speaking router to select multiple iBGP paths as the best paths to a destination. The best paths or multipaths are then installed in the IP routing table of the router.

Per-vrf label mode is not supported for Carrier Supporting Carrier (CSC) network with internal and external BGP multipath setup.

Per-VRF label mode cannot be used for BGP PIC edge with eBGP multipath, as that might cause loops. Only per-prefix label supports per-VRF label mode.

Per-VRF label mode does not support BGP best external within the same address family, as it may cause loops during re-convergence scenarios.

Configure iBGP Multipath Load Sharing

Perform this task to configure the iBGP Multipath Load Sharing:

SUMMARY STEPS

1. configure

2. **router bgp** *as-number*
3. **address-family** {*ipv4|ipv6*} {*unicast|multicast*}
4. **maximum-paths ibgp** *number*
5. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

Step 1 **configure****Example:**

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number***Example:**

```
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 3 **address-family** {*ipv4|ipv6*} {*unicast|multicast*}**Example:**

```
Router(config-bgp)# address-family ipv4 multicast
```

Specifies either the IPv4 or IPv6 address family and enters address family configuration submenu.

Step 4 **maximum-paths ibgp** *number***Example:**

```
Router(config-bgp-af)# maximum-paths ibgp 30
```

Configures the maximum number of iBGP paths for load sharing.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
 - **No** —Exits the configuration session without committing the configuration changes.
 - **Cancel** —Remains in the configuration session, without committing the configuration changes.
-

iBGP Multipath Loadsharing Configuration: Example

The following is a sample configuration where 30 paths are used for loadsharing:

```
router bgp 100
  address-family ipv4 multicast
    maximum-paths ibgp 30
  !
!
end
```

Accumulated IGP Attribute for BGP

Table 22: Feature History Table

Feature Name	Release Information	Feature Description
Accumulated IGP Attribute for BGP	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *Accumulated IGP Attribute for BGP is now supported on Cisco 8712-MOD-M routers.
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.

Overview of BGP AIGP

The Accumulated IGP (AIGP) Attribute for BGP is an optional non-transitive BGP path Attribute. IANA assigned the attribute type code for the AIGP attribute. The value field of the AIGP attribute is defined as a set of Type/Length/Value elements (TLVs). The AIGP TLV contains the Accumulated IGP metric.

The AIGP feature is required in the network to simulate the current OSPF behavior of computing the distance associated with a path. OSPF or LDP carries the prefix or label information only in the local area. Then, BGP carries the prefix label to all the remote areas by redistributing the routes into BGP at area boundaries. The routes or labels are then advertised using LSPs. The next hop for the route is changed at each ABR to local router which removes the need to leak OSPF routes across area boundaries. The bandwidth available on each of the core links is mapped to OSPF cost, hence it is imperative that BGP carries this cost correctly between each of the PEs. This functionality is achieved by using the AIGP.

Originate Prefixes with AIGP

Origination of routes with the accumulated interior gateway protocol (AIGP) metric is controlled by configuration. AIGP attributes are attached to redistributed routes that satisfy following conditions.

- The protocol redistributing the route is enabled for AIGP.

- The route is an interior gateway protocol (IGP) route redistributed into border gateway protocol (BGP). The value assigned to the AIGP attribute is the value of iGP next hop to the route or as set by a route-policy.
- The route is a static route redistributed into BGP. The value assigned is the value of next hop to the route or as set by a route-policy.
- The route is imported into BGP through network statement. The value assigned is the value of next hop to the route or as set by a route-policy.

Configuration Examples

Originate prefixes with AIGP.

```
Router(config)# route-policy aip_policy
Router(config-rpl)# set aigp-metric igp-cost
Router(config-rpl)# exit
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# redistribute ospf route-policy aip_policy
```

Running Configuration

```
route-policy aip_policy
  set aigp-metric igp-cost
!
router bgp 100
  address-family ipv4 unicast
    redistribute ospf route-policy aip_policy
```

Verification

Verify the status of the AIGP attribute.

```
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.
```

Accumulated Interior Gateway Protocol Attribute

The Accumulated Interior Gateway Protocol (AiGP) Attribute is an optional non-transitive BGP Path Attribute. The attribute type code for the AiGP Attribute is to be assigned by IANA. The value field of the AiGP Attribute is defined as a set of Type/Length/Value elements (TLVs). The AiGP TLV contains the Accumulated IGP Metric.

The AiGP feature is required in the 3107 network to simulate the current OSPF behavior of computing the distance associated with a path. OSPF/LDP carries the prefix/label information only in the local area. Then, BGP carries the prefix/label to all the remote areas by redistributing the routes into BGP at area boundaries. The routes/labels are then advertised using LSPs. The next hop for the route is changed at each ABR to local router which removes the need to leak OSPF routes across area boundaries. The bandwidth available on each of the core links is mapped to OSPF cost, hence it is imperative that BGP carries this cost correctly between each of the PEs. This functionality is achieved by using the AiGP.

BGP Accept Own

The BGP Accept Own feature enables handling of self-originated VPN routes, which a BGP speaker receives from a route-reflector (RR). A "self-originated" route is one which was originally advertised by the speaker itself. As per BGP protocol [RFC4271], a BGP speaker rejects advertisements that were originated by the speaker itself. However, the BGP Accept Own mechanism enables a router to accept the prefixes it has advertised, when reflected from a route-reflector that modifies certain attributes of the prefix. A special community called ACCEPT-OWN is attached to the prefix by the route-reflector, which is a signal to the receiving router to bypass the ORIGINATOR_ID and NEXTHOP/MP_REACH_NLRI check. Generally, the BGP speaker detects prefixes that are self-originated through the self-origination check (ORIGINATOR_ID, NEXTHOP/MP_REACH_NLRI) and drops the received updates. However, with the Accept Own community present in the update, the BGP speaker handles the route.

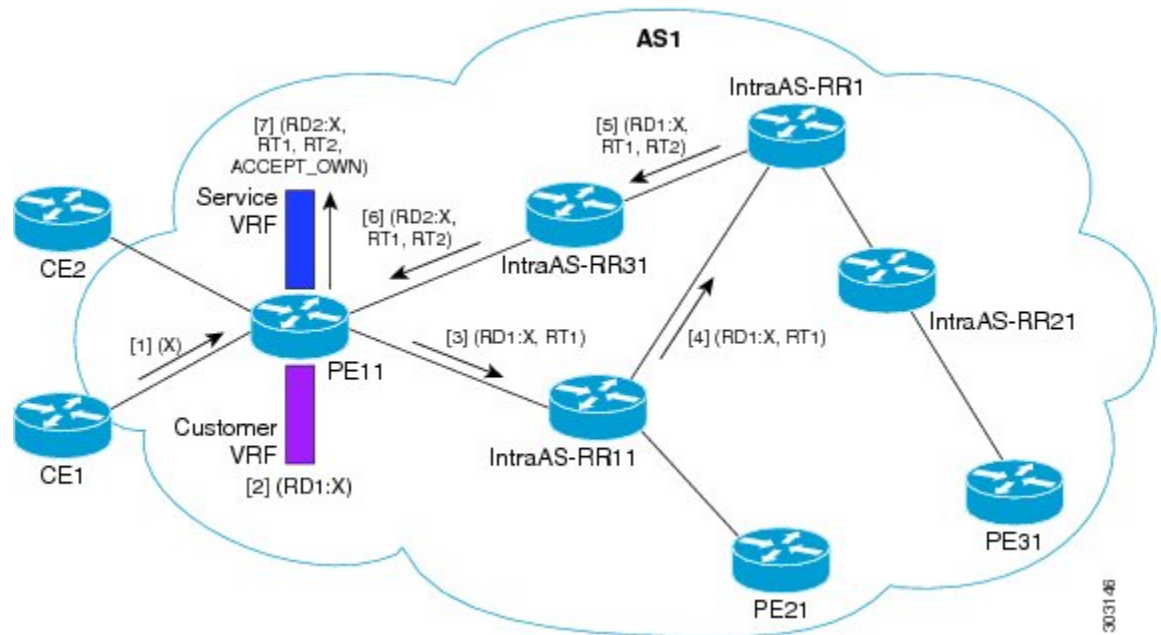
One of the applications of BGP Accept Own is auto-configuration of extranets within MPLS VPN networks. In an extranet configuration, routes present in one VRF is imported into another VRF on the same PE. Normally, the extranet mechanism requires that either the import-rt or the import policy of the extranet VRFs be modified to control import of the prefixes from another VRF. However, with Accept Own feature, the route-reflector can assert that control without the need for any configuration change on the PE. This way, the Accept Own feature provides a centralized mechanism for administering control of route imports between different VRFs.

BGP Accept Own is supported only for VPNv4 and VPNv6 address families in neighbor configuration mode.

Route-Reflector Handling Accept Own Community and RTs

The ACCEPT_OWN community is originated by the InterAS route-reflector (InterAS-RR) using an outbound route-policy. To minimize the propagation of prefixes with the ACCEPT_OWN community attribute, the attribute will be attached on the InterAS-RR using an outbound route-policy towards the originating PE. The InterAS-RR adds the ACCEPT-OWN community and modifies the set of RTs before sending the new Accept Own route to the attached PEs, including the originator, through intervening RRs. The route is modified via route-policy.

Accept Own Configuration Example



In this configuration example:

- PE11 is configured with Customer VRF and Service VRF.
- OSPF is used as the IGP.
- VPNv4 unicast and VPNv6 unicast address families are enabled between the PE and RR neighbors and IPv4 and IPv6 are enabled between PE and CE neighbors.

The Accept Own configuration works as follows:

1. CE1 originates prefix X.
2. Prefix X is installed in customer VRF as (RD1:X).
3. Prefix X is advertised to IntraAS-RR11 as (RD1:X, RT1).
4. IntraAS-RR11 advertises X to InterAS-RR1 as (RD1:X, RT1).
5. InterAS-RR1 attaches RT2 to prefix X on the inbound and ACCEPT_OWN community on the outbound and advertises prefix X to IntraAS-RR31.
6. IntraAS-RR31 advertises X to PE11.
7. PE11 installs X in Service VRF as (RD2:X, RT1, RT2, ACCEPT_OWN).

Remote PE: Handling of Accept Own Routes

Remote PEs (PEs other than the originator PE), perform bestpath calculation among all the comparable routes. The bestpath algorithm has been modified to prefer an Accept Own path over non-Accept Own path. The bestpath comparison occurs immediately before the IGP metric comparison. If the remote PE receives an Accept Own path from route-reflector 1 and a non-Accept Own path from route-reflector 2, and if the paths are otherwise identical, the Accept Own path is preferred. The import operates on the Accept Own path.

Configuring BGP Accept Own

Perform this task to configure BGP Accept Own:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **update-source** *type interface-path-id*
6. **address-family** {*vpn4 unicast* | *vpn6 unicast*}
7. **accept-own** [**inheritance-disable**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure Example: RP/0/RP0/CPU0:router# configure	Enters mode.
Step 2	router bgp <i>as-number</i> Example: 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 3	neighbor <i>ip-address</i> Example: Router(config-bgp)#neighbor 10.1.2.3	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	remote-as <i>as-number</i> Example: Router(config-bgp-nbr)#remote-as 100	Assigns a remote autonomous system number to the neighbor.
Step 5	update-source <i>type interface-path-id</i> Example: Router(config-bgp-nbr)#update-source Loopback0	Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.
Step 6	address-family { <i>vpn4 unicast</i> <i>vpn6 unicast</i> } Example: Router(config-bgp-nbr)#address-family vpnv6 unicast	Specifies the address family as VPNv4 or VPNv6 and enters neighbor address family configuration mode.
Step 7	accept-own [inheritance-disable] Example:	Enables handling of self-originated VPN routes containing Accept_Own community.

	Command or Action	Purpose
	<code>Router(config-bgp-nbr-af) #accept-own</code>	Use the inheritance-disable keyword to disable the "accept own" configuration and to prevent inheritance of "acceptown" from a parent configuration.

BGP Link-State

Table 23: Feature History Table

Feature Name	Release Information	Feature Description
BGP Link-State	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700) (select variants only*)</p> <p>This feature allows efficient sharing of IGP link-state information through BGP, improving network topology awareness for applications like Segment Routing Path Computation Element (SR-PCE). It uses Address Family Identifier (AFI) and Sub-address Family Identifier (SAFI) to encode link-state data in the BGP-LS attribute, as per RFC7752. This functionality enables SR-PCE to learn node capabilities and compute paths using Segment Routing Traffic Engineering (SR-TE).</p> <p>*BGP Link-State functionality is now supported on Cisco-8712-MOD-M routers.</p>

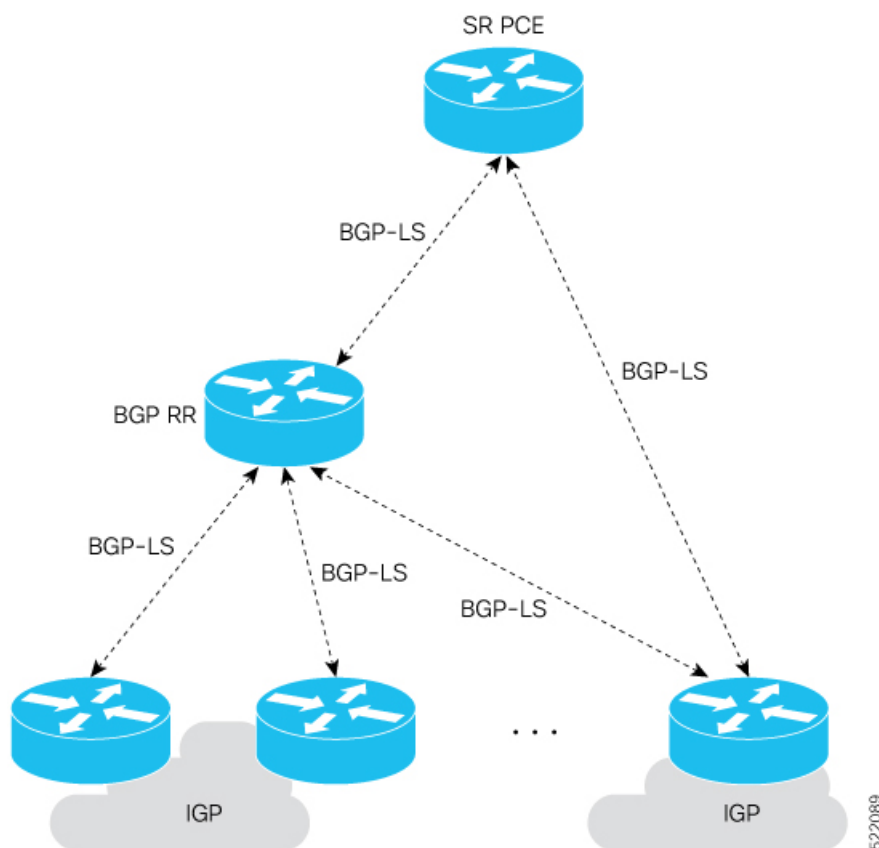
BGP Link-State (LS) is an Address Family Identifier (AFI) and Sub-address Family Identifier (SAFI) originally defined to carry interior gateway protocol (IGP) link-state information through BGP. The BGP Network Layer Reachability Information (NLRI) encoding format for BGP-LS and a new BGP Path Attribute called the BGP-LS attribute are defined in [RFC7752](#). The identifying key of each Link-State object, namely a node, link, or prefix, is encoded in the NLRI and the properties of the object are encoded in the BGP-LS attribute.



Note IGP's do not use BGP LS data from remote peers. BGP does not download the received BGP LS data to any other component on the router.

An example of a BGP-LS application is the Segment Routing Path Computation Element (SR-PCE). The SR-PCE can learn the SR capabilities of the nodes in the topology and the mapping of SR segments to those nodes. This can enable the SR-PCE to perform path computations based on SR-TE and to steer traffic on paths different from the underlying IGP-based distributed best-path computation.

The following figure shows a typical deployment scenario. In each IGP area, one or more nodes (BGP speakers) are configured with BGP-LS. These BGP speakers form an iBGP mesh by connecting to one or more route-reflectors. This way, all BGP speakers (specifically the route-reflectors) obtain Link-State information from all IGP areas (and from other ASes from eBGP peers).



Exchange Link State Information with BGP Neighbor

The following example shows how to exchange link-state information with a BGP neighbor:

```
Router# configure
Router(config)# router bgp 1
Router(config-bgp)# neighbor 10.0.0.2
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# address-family link-state link-state
Router(config-bgp-nbr-af)# exit
```

IGP Link-State Database Distribution

A given BGP node may have connections to multiple, independent routing domains. IGP link-state database distribution into BGP-LS is supported for both OSPF and IS-IS protocols in order to distribute this information on to controllers or applications that desire to build paths spanning or including these multiple domains.

To distribute OSPFv2 link-state data using BGP-LS, use the **distribute link-state** command in router configuration mode.

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

Usage Guidelines and Limitations

- BGP-LS supports IS-IS and OSPFv2.
- The identifier field of BGP-LS (referred to as the Instance-ID) identifies the IGP routing domain where the NLRI belongs. The NLRIs representing link-state objects (nodes, links, or prefixes) from the same IGP routing instance must use the same Instance-ID value.
- When there is only a single protocol instance in the network where BGP-LS is operational, we recommend configuring the Instance-ID value to 0.
- Assign consistent BGP-LS Instance-ID values on all BGP-LS Producers within a given IGP domain.
- NLRIs with different Instance-ID values are considered to be from different IGP routing instances.
- Unique Instance-ID values must be assigned to routing protocol instances operating in different IGP domains. This allows the BGP-LS Consumer (for example, SR-PCE) to build an accurate segregated multi-domain topology based on the Instance-ID values, even when the topology is advertised via BGP-LS by multiple BGP-LS Producers in the network.
- If the BGP-LS Instance-ID configuration guidelines are not followed, a BGP-LS Consumer may see duplicate link-state objects for the same node, link, or prefix when there are multiple BGP-LS Producers deployed. This may also result in the BGP-LS Consumers getting an inaccurate network-wide topology.

Configuring BGP Link-state

To exchange BGP link-state (LS) information with a BGP neighbor, perform these steps:

Procedure

Step 1 **configure****Example:**

```
RP/0/RP0/CPU0:router# configure
Enters mode.
```

Step 2 **router bgp** *as-number***Example:**

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

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address***Example:**

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

Configures a CE neighbor. The ip-address argument must be a private address.

Step 4 **remote-as** *as-number***Example:**

```
Router(config-bgp-nbr)# remote-as 1
```

Configures the remote AS for the CE neighbor.

Step 5 **address-family link-state link-state****Example:**

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

Distributes BGP link-state information to the specified neighbor.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Domain Distinguisher

To configure unique identifier four-octet ASN, perform these steps:

Procedure

Step 1 **configure****Example:**

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number***Example:**

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

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family link-state link-state**

Example:

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

Enters address-family link-state configuration mode.

Step 4 **domain-distinguisher** *unique-id***Example:**

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

Configures unique identifier four-octet ASN. Range is from 1 to 4294967295.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

BGP Permanent Network

BGP permanent network feature supports static routing through BGP. BGP routes to IPv4 or IPv6 destinations (identified by a route-policy) can be administratively created and selectively advertised to BGP peers. These routes remain in the routing table until they are administratively removed. A permanent network is used to define a set of prefixes as permanent, that is, there is only one BGP advertisement or withdrawal in upstream for a set of prefixes. For each network in the prefix-set, a BGP permanent path is created and treated as less preferred than the other BGP paths received from its peer. The BGP permanent path is downloaded into RIB when it is the best-path.

The **permanent-network** command in global address family configuration mode uses a route-policy to identify the set of prefixes (networks) for which permanent paths is to be configured. The **advertise permanent-network** command in neighbor address-family configuration mode is used to identify the peers to whom the permanent paths must be advertised. The permanent paths is always advertised to peers having the advertise permanent-network configuration, even if a different best-path is available. The permanent path is not advertised to peers that are not configured to receive permanent path.

The permanent network feature supports only prefixes in IPv4 unicast and IPv6 unicast address-families under the default Virtual Routing and Forwarding (VRF).

Restrictions

These restrictions apply while configuring the permanent network:

- Permanent network prefixes must be specified by the route-policy on the global address family.
- You must configure the permanent network with route-policy in global address family configuration mode and then configure it on the neighbor address family configuration mode.

- When removing the permanent network configuration, remove the configuration in the neighbor address family configuration mode and then remove it from the global address family configuration mode.

Configuring BGP Permanent Network

Perform this task to configure BGP permanent network. You must configure at least one route-policy to identify the set of prefixes (networks) for which the permanent network (path) is to be configured.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **prefix-set** *prefix-set-name*

Example:

```
Router(config)# prefix-set PERMANENT-NETWORK-IPv4
Router(config-pfx)# 1.1.1.1/32,
Router(config-pfx)# 2.2.2.2/32,
Router(config-pfx)# 3.3.3.3/32
Router(config-pfx)# end-set
```

Enters prefix set configuration mode and defines a prefix set for contiguous and non-contiguous set of bits.

Step 3 **exit**

Example:

```
Router(config-pfx)# exit
```

Exits prefix set configuration mode and enters global configuration mode.

Step 4 **route-policy** *route-policy-name*

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
```

Creates a route policy and enters route policy configuration mode, where you can define the route policy.

Step 5 **end-policy**

Example:

```
Router(config-rpl)# end-policy
```


Ends the definition of a route policy and exits route policy configuration mode.

Step 6 **router bgp** *as-number*

Example:

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

Specifies the autonomous system number and enters the BGP configuration mode.

Step 7 **address-family { ipv4 | ipv6 } unicast**

Example:

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

Step 8 **permanent-network route-policy** *route-policy-name*

Example:

```
Router(config-bgp-af)# permanent-network route-policy POLICY-PERMANENT-NETWORK-IPv4
```

Configures the permanent network (path) for the set of prefixes as defined in the route-policy.

Step 9 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 10 **show bgp {ipv4 | ipv6} unicast** *prefix-set*

Example:

```
show bgp ipv4 unicast
```

(Optional) Displays whether the prefix-set is a permanent network in BGP.

Advertise Permanent Network

Perform this task to identify the peers to whom the permanent paths must be advertised.

Procedure

Step 1 **configure****Example:**

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number***Example:**

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

Specifies the autonomous system number and enters the BGP configuration mode.

Step 3 **neighbor** *ip-address***Example:**

```
Router(config-bgp)# neighbor 10.255.255.254
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **remote-as** *as-number***Example:**

```
Router(config-bgp-nbr)# remote-as 4713
```

Assigns the neighbor a remote autonomous system number.

Step 5 **address-family { ipv4 | ipv6 } unicast****Example:**

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

Step 6 **advertise permanent-network****Example:**

```
Router(config-bgp-nbr-af)# advertise permanent-network
```

Specifies the peers to whom the permanent network (path) is advertised.

Step 7 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 8 **show bgp {ipv4 | ipv6} unicast neighbor** *ip-address*

Example:

```
Router# show bgp ipv4 unicast neighbor 10.255.255.254
```

(Optional) Displays whether the neighbor is capable of receiving BGP permanent networks.

Selective FIB Download

Table 24: Feature History Table

Feature Name	Release Information	Feature Description
Selective FIB Download	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8200, 8700); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q200])</p> <p>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.</p>

Network challenges

In a network setup with multiple data centers, network links may go down, affecting routing behavior. When connections between intermediate distribution facilities (IDFs) and their termination points are unavailable, these IDFs lack precise routing paths to a target location.

Traffic management

A main routing node manages traffic by distributing it across available routes based on anchor routes. (An anchor route is an aggregated network prefix used within a specific network layer by consolidating multiple specific routes into a single broader prefix.) Some traffic may pass through an intermediate node, guided by anchor routes from the affected IDFs. If the intermediate node has these anchor routes installed, it forwards the traffic to the IDFs.

Routing issues

When traffic reaches the IDFs, the absence of specific destination routes can cause issues. If anchor routes are present, traffic goes to a null interface, creating a black hole where traffic is discarded.

Solution: Selective FIB download

Selective FIB download feature ensures that local nodes and connected intermediate nodes do not install anchor routes in their Routing Information Base (RIB). Instead, if direct routes to specific network destinations are not available, they use default routes. This setup allows traffic to follow default routes to alternative nodes when anchor routes are not installed on intermediate nodes. These alternative nodes, which have specific routes from other IDFs, forward the traffic through a series of nodes until it reaches its termination point and returns to the main routing node, thus avoiding traffic black holes.

Address-Family Support

This feature supports prefixes in IPv4 unicast and IPv6 unicast address-families under default VRF.

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 (download) 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.

The following sample outputs illustrates 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 Mechanism for Update Generation

The Border Gateway Protocol-Routing Information Base (BGP-RIB) feedback mechanism for update generation feature avoids premature route advertisements and subsequent packet loss in a network. This mechanism ensures that routes are installed locally, before they are advertised to a neighbor.

BGP waits for feedback from RIB indicating that the routes that BGP installed in RIB are installed in forwarding information base (FIB) before BGP sends out updates to the neighbors. RIB uses the the BCDL feedback mechanism to determine which version of the routes have been consumed by FIB, and updates the BGP with that version. BGP will send out updates of only those routes that have versions up to the version that FIB has installed. This selective update ensures that BGP does not send out premature updates resulting in attracting traffic even before the data plane is programmed after router reload, LC OIR, or flap of a link where an alternate path is made available.

To configure BGP to wait for feedback from RIB indicating that the routes that BGP installed in RIB are installed in FIB, before BGP sends out updates to neighbors, use the **update wait-install** command in router address-family IPv4 or router address-family VPNv4 configuration mode. The **show bgp**, **show bgp neighbors**, and **show bgp process performance-statistics** commands display the information from update wait-install configuration.

Delay BGP Route Advertisements

Table 25: Feature History Table

Feature Name	Release Information	Feature Description
Delay BGP Route Advertisements	Release 7.5.3	<p>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.</p> <p>You can configure a minimum delay of 1 second and a maximum delay of 600 seconds.</p> <p>This feature introduces the update wait-install delay startup command.</p>

When BGP forwards traffic, it waits for feedback from the RIB until the RIB is ready to forward traffic. Once the RIB is ready, BGP sends the route updates to the BGP neighbors and peer-groups. Advertising routes before the RIB is synchronized in the FIB results in traffic loss. To avoid this problem, the router must delay the BGP start-up process to delay the BGP update generation so that no traffic loss happens.

To accomplish this, you must configure the **update wait-install delay startup** command to delay the generation of BGP updates. The **show bgp process** command displays the delay of the BGP process update since the last router reload.

This feature allows you to configure the minimum and maximum delay periods. The range of the delay is from 1 second to 600 seconds. As a result, network traffic loss is avoided.

Restrictions

This feature is applicable for the following Address Family Indicators (AFIs):

- IPv4 unicast
- IPv6 unicast
- VPNv4 unicast
- VPNv6 unicast

Configuration

1. Enter the IOS XR configuration mode.

```
Router# configure
```

2. Specify the BGP Autonomous System Number (AS Number).

```
Router(config)# router bgp 1
```

3. Specify the IP address from the address-family (Pv4, IPv6, VPNv4, or VPNv6) options.

```
Router(config-bgp)# address-family {ipv4| ipv6| vpnv4| vpn6} unicast
```

For example,

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

4. Schedule the delay of the BGP process to prevent routes from being advertised to peers until RIB is synchronized.

```
Router(config-bgp-af)# update wait-install delay startup (time in seconds)
```

For example,

```
Router(config-bgp-af)# update wait-install delay startup 10
```

5. Commit the changes.

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



Note The delay time ranges from 1 second to 600 seconds.

Running Configuration

```
configure
router bgp 1
  address-family ipv4 unicast
    update wait-install delay startup 10
!
```

Verification Example

The following command displays the delay of the BGP process update:

```
Router# show running-config router bgp 1
router bgp 1
  address-family ipv4 unicast
    update wait-install delay startup 10
```

What to do next

You can then run the `show bgp process` command. The **Update wait-install enabled** section in the `show bgp process` command displays the delay of the BGP process update since the last router reload.

```
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

--More--
```

Default-originate Under VRF

BGP advertises default routes to provider-edge neighbors, based on per-VRF configuration.

User-Defined Martian Address Check

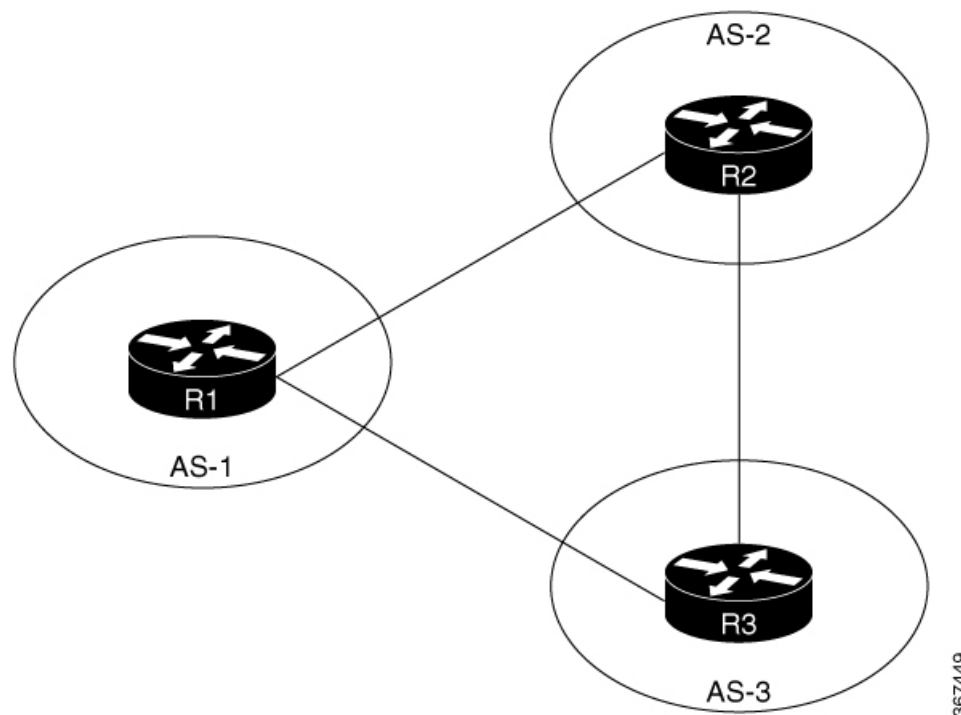
When you configure BGP on a Cisco 8000 Series Router, you can prevent routers from accessing certain sites with certain IP address prefixes. These routers drop packets from such IP addresses, and such IP addresses are known as Martian addresses. However, you can enable routers with BGP IPv4 address-family or BGP IPv6 address-family configuration to access these sites by configuring the command **default-martian-check disable**. These sites are sites with certain IPv4 and IPv6 prefixes as follows:

- IPv4 address prefixes
 - 0.0.0.0/8
 - 127.0.0.0/8
 - 224.0.0.0/4
- IPv6 address prefixes
 - ::
 - ::0002 - ::ffff
 - ::ffff:a.b.c.d
 - fe80:xxxx
 - ffxx:xxxx

- The ability to ignore as-path onwards while computing multipath is added. The **bgp multipath as-path ignore onwards** command ignores as-path onwards while computing multipath.

When multiple connected routers start ignoring as-path onwards while computing multipath, it causes routing loops. Therefore, you should not configure the **bgp multipath as-path ignore onwards** command on routers that can form a loop.

Figure 12: Topology to illustrate formation of loops



Consider three routers R1, R2 and R3 in different autonomous systems (AS-1, AS-2, and AS-3). The routers are connected with each other. R1 announces a prefix to R2 and R3. Both R2 and R3 are configured with multipath and also with **bgp multipath as-path ignore onwards** command. Since R3 is configured as multipath, R2 will send part of its traffic to R3. Similarly, R3 will send part of its traffic to R2. This creates a forwarding loop between R3 and R2. Therefore, to avoid such forwarding loops you should not configure the **bgp multipath as-path ignore onwards** command on connected routers.

Overview of BGP Monitoring Protocol

The BGP Monitoring Protocol (BMP) feature enables monitoring of BGP speakers (called BMP clients). You can configure a device to function as a BMP server, which monitors either one or several BMP clients, which in turn, has several active peer sessions configured. You can also configure a BMP client to connect to one or more BMP servers. The BMP feature enables configuration of multiple BMP servers (configured as primary servers) to function actively and independent of each other, simultaneously to monitor BMP clients.

The BMP Protocol provides access to the Adjacent Routing Information Base, Incoming (Adj-RIB-In) table of a peer on an ongoing basis and a periodic dump of certain statistics that the monitoring station can use for further analysis. The BMP provides pre-policy view of the Adj-RIB-In table of a peer.

There can be several BMP servers configured globally across all the BGP instances. The BMP servers configured are common across multiple speaker instances and each BGP peer in an instance can be configured for

monitoring by all or a subset of the BMP servers, giving a 'any-to-any' map between BGP peers and BMP servers from the point of view of a BGP speaker. If a BMP server is configured before any of the BGP peers come up, then the monitoring will start as soon as the BGP peers come up. A BMP server configuration can be removed only when there are no BGP peers configured to be monitored by that particular BMP server.

Sessions between BMP clients and BMP servers operate over plain TCP (no encryption/encapsulation). If a TCP session with the BMP server is not established, the client retries to connect every 7 seconds.

The BMP server does not send any messages to its clients (BGP speakers). The message flow is in one direction only—from BGP speakers to the BMP servers

A maximum of eight BMP servers can be configured on the router. Each BMP server is specified by a server ID and certain parameters such as IP address, port number, etc are configurable. Upon successful configuration of a BMP server with host and port details, the BGP speaker attempts to connect to BMP Server. Once the TCP connection is setup, an Initiation message is sent as first message.

The **bmp server** command enables the user to configure multiple—independent and asynchronous—BMP server connections.

All neighbors for a BGP speaker need not necessarily be BMP clients. BMP clients are the ones that have direct TCP connection with a BMP server. Each of these BGP speakers can have many BGP neighbors or peers. Under a BGP speaker, if any of its neighbors are configured for BMP monitoring, only that particular peer router's messages are sent to BMP servers.

The session connection to BMP server is attempted after an initial-delay at the BMP client. This initial-delay can be configured. If the initial-delay is not configured, then the default connection delay of 7 seconds is used. Configuring the initial delay becomes significant under certain circumstances where, if multiple BMP servers' states toggle closely and refresh delay is so small, then this might result in redundant route-refreshes being generated. This causes considerable network traffic and load on the device. Having different initial delays can reduce the load spike on the network and router.

After the initial delay, TCP connection to BMP servers are attempted. Once the server connections are up, it is checked if there are any peers enabled for monitoring. Once a BGP peer that is already being monitored is in the "ESTAB" state, speaker sends a "peer-up" message for that peer to the BMP server. After the BGP peer receives a route-refresh request, neighbor sends the updates. This route refresh is initiated based on a delay configured for each BMP server. This is called route refresh delay. When there are multiple neighbors to be monitored, each neighbor is set a refresh delay based upon the BMP server they are enabled for. Once all the BGP neighbors have sent the updates in response to the refresh requests, the tables will be up to date in the BMP Server. If a neighbor establishes connection after BMP monitoring has begun, it does not require a route-refresh request. All received routes from that neighbor is sent to BMP servers.



Note In the case of BMP Pre Inbound Policy Route monitoring, when a new BMP server comes up, route refresh requests are sent to the peer router by the BGP speaker. However, in the case of BMP Post Inbound Policy Route Monitoring route refresh request are not sent to the peer routers when the new BMP server comes up because the BMP table is used for update generation.

It is advantageous to batch up refresh requests to BGP peers, if several BMP servers are activated in quick succession. Use the **bmp server initial-refresh-delay** command to configure a delay in triggering the refresh mechanism when the first BMP server comes up. If other BMP servers come online within this time-frame, only one set of refresh requests is sent to the BGP peers. You can also configure the **bmp server initial-refresh-delay skip** command to skip all refresh requests from BGP speakers and just monitor all incoming messages from the peers.

In a client-server configuration, it is recommended that the resource load of the devices be kept minimal and adding excessive network traffic must be avoided. In the BMP configuration, you can configure various delay timers on the BMP server to avoid flapping during connection between the server and client.

Adj-RIB-In Post-Policy View for L3VPN Address Families

Table 26: Feature History Table

Feature Name	Release	Description
Adj-RIB-In Post-Policy View for L3VPN Address Families	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.
Adj-RIB-In Post-Policy View for L3VPN Address Families	Release 7.5.4	After applying policy filters, you can now monitor BGP events and collect BGP route information and statistics for L3VPN address families for unprocessed routing information. This is made possible because this feature enables the BGP Monitoring Protocol (BMP) to allow a BGP router to advertise the BGP Adj-RIB-In post-policy for L3VPN address families. This feature introduces these changes: <ul style="list-style-type: none"> • CLI: This feature modifies the following commands: <ul style="list-style-type: none"> • show bgp bmp • route-monitoring inbound post-policy • YANG Data Model: New XPaths for <ul style="list-style-type: none"> • Cisco-IOS-Xr-bgp-adj-rib-in-post-policy.yang • Cisco-IOS-Xr-rip4-hop-quer.yang (see GitHub, YANG Data Models Navigator)

The BGP Monitoring Protocol (BMP), defined in [RFC 7854](#), is a protocol to monitor BGP events as well as BGP route information and statistics. Using this protocol, a BMP collector can monitor various routing

information bases within a BGP speaker such as Adj-RIB-In (Pre-Policy and Post-Policy), Local RIB and Adj-RIB-Out (Pre-Policy and Post-Policy). This provides comprehensive insights into real-time and historical operation of a BGP network which can be used for route monitoring, routing analytics, and traffic engineering analytics. BMP can additionally send information on peer state change events, including why a peer went down in the case of a BGP event.

The Adj-RIB-In pre-policy (also referred to as Inbound pre-policy) conveys to a BMP receiver all unprocessed routing information that has been advertised to the local BGP speaker by its peers before any inbound policy has been applied. The Adj-RIB-In post-policy (also referred to as Inbound post-policy) conveys to a BMP receiver all routing information after policy filters and/or modifications (such as addition or deletion of BGP attributes) have been applied.

BMP provides access to the Adjacent Routing Information Base - Inbound (Adj-RIB-In) table of a peer on an ongoing basis and statistics that the monitoring station can use for further analysis. BMP allows a BGP router to advertise the pre-policy or post-policy BGP Adj-RIB-In from the specific BGP peers to a monitoring station.

BGP Adj-RIB-In post-policy (inbound post-policy) view for L3VPN traffic shows the routing information that a BGP peer gets from another peer BGP speaker after applying a BGP input policy and exports the route information to BMP server. The policy instructs the router to inspect routes, filter them, and potentially modify their attributes as they are accepted from a peer, advertised to a peer, or redistributed from one routing protocol to another.

To enable the Adj-RIB-In post-policy (inbound post-policy) for L3VPN address families, you must run configure the **route-monitoring inbound post-policy** command.

In addition to the existing RIB views available for monitoring (see [Overview of BGP Monitoring Protocol, on page 132](#)), Cisco IOS XR Release 7.5.4 adds the following address families in the Adj-RIB-In Post-Policy view for monitoring L3VPN BGP network:

- Default VRF
 - VPNv4 Unicast
 - VPNv6 Unicast
- Non-Default VRF
 - IPv4 Unicast
 - IPv6 Unicast

Configuration

Configure the **route-monitoring inbound post-policy** command to enable the Adj-RIB-In post-policy (inbound post-policy) view by performing the following actions:

```
Router# config
Router(config)#bmp server all
Router(config-bgp-bmp)#route-monitoring inbound post-policy
Router(config-bgp-bmp-rmon)#commit
```

Running Configuration

```
bmp server all
route-monitoring inbound post-policy
```

!
!

Verification

Verify whether the Adj-RIB-In post-policy (inbound post-policy) configuration is done by running the **show bgp bmp server <server ID>** command.

```
Router# show bgp bmp server 1
Tue Nov 29 19:02:27.837 IST
BMP server 1
Host 12.1.2.1 Port 16001
Connected for 05:51:09
Last Disconnect event received : 00:00:00
Precedence: internet
BGP neighbors: 7
VRF: - (0x60000000)
Update Source: - (-)
Update Source Vrf ID: 0x0
Update Mode                               : In-Post-Policy
  In-Post-Policy
    Advertisement interval                 : 15 secs
    Scanner interval                       : 60 secs
  Flapping Delay                          : 300 secs
  Initial Delay                           : 0 secs
  Initial Refresh Delay                    : 1 secs
  Initial Refresh Spread                   : 1 secs
  Stats Reporting Period                   : 0 secs
  Queue Route Mon Msg buffer limit         : 133693 KB (Current Server Up Count: 2)
  Queue Route Mon Msg buffer usage         : 0 B
  Queue write pulse sent                   : Nov 29 13:13:15.484, Nov 29 13:11:53.478 (all)
  Queue write pulse received               : Nov 29 13:13:15.484
  Update Generation in Progress            : No
  Reset Walk in Progress                   : No
-----More-----
```

You can then configure the following commands:

- **bmp advertisement-interval** to set the minimum interval between the sending of BMP routing updates.
- **bmp scan-time** to configure scanning intervals of BMP-speaking networking devices.

Local-RIB view for IP and L3VPN Address Families

Table 27: Feature History Table

Feature Name	Release	Description
Local-RIB view for IP and L3VPN Address Families	Release 7.5.4	<p>After applying policy filters, you can now monitor BGP events and collect BGP best path information and statistics for IP and L3VPN address families for unprocessed routing information.</p> <p>This is made possible because this feature enables BMP to allow a BGP router to advertise the BGP Local-RIB for IP and L3VPN address families.</p> <p>Operators may wish to validate the impact of policies applied to the Adj-RIB-In by analysing the final decision made by the router when installing into the Loc-RIB.</p> <p>This feature introduces these changes:</p> <ul style="list-style-type: none"> • CLI: Modifies the show bgp bmp command. • YANG Data Model: New XPaths for <ul style="list-style-type: none"> • <code>Cisco-IOS-XR-adj-rbp-obj.yang</code> • <code>Cisco-IOS-XR-ip4-bgp-obj.yang</code> (see GitHub, YANG Data Models Navigator)

The Local-RIB (Loc-RIB) contains the routes that are received from the BGP peers and selected by the local BGP speaker's decision process. The Adj-RIB-In may contain hundreds of thousands of routes for per peer. But only a few of routes are selected and installed in the Loc-RIB after the best-path selection.

The Loc-RIB contains the routes selected by the local BGP speaker's Decision Process and are considered valid to it.

For example, the Adj-RIB-In for a given peer post-policy (inbound post-policy) may contain thousands of routes per peer. But only a few of routes are selected and installed in the Loc-RIB after the best-path selection.

The monitoring application that requires to correlate flow records to Loc-RIB entries, needs to collect and monitor the routes that are actually selected and used. The Loc-RIB includes all selected received routes from BGP peers in addition to locally originated routes. It also contains the address family, the prefixes, attributes, and prefixes for address families.

Starting from Release 7.5.4, the Loc-RIB view (best-path only) is available for monitoring for the following address families:

- Default VRF
 - IPv4 Unicast
 - IPv4 Labeled Unicast
 - IPv6 Unicast
 - IPv6 Labeled Unicast
 - VPNv4 Unicast
 - VPNv6 Unicast
- Non-Default VRF
 - IPv4 Unicast
 - IPv6 Unicast

This feature complies with [RFC 9069](#).

Configuration

Configure the **route-monitoring local-rib** command to enable the local-RIB view by performing the following actions:

```
Router# config
Router(config)#bmp server all
Router(config-bgp-bmp)#route-monitoring local-rib
Router(config-bgp-bmp-rmon)#commit
```

Running Configuration

```
bmp server all
  route-monitoring local-rib
  !
  !
```

Verification

Verify whether the Local RIB (Loc-RIB) configuration is done by running the **show bgp bmp server <server ID>** command.

```
Router#show bgp bmp server 1
BMP server 1
Host 12.1.2.1 Port 16001
Connected for 06:00:39
Last Disconnect event received : 00:00:00
Precedence: internet
BGP neighbors: 10
VRF: - (0x60000000)
Update Source: - (-)
Update Source Vrf ID: 0x0
Update Mode : In-Post-Policy, Local-RIB
In-Post-Policy
```



```

Advertisement interval      : 15 secs
Scanner interval          : 60 secs
Local-RIB
Advertisement interval      : 15 secs
Scanner interval          : 60 secs
  Global                   : 60 secs
  IPv4 Unicast              : 60 secs
  VPNv4 Unicast             : 60 secs
  IPv6 Unicast              : 60 secs
  VPNv6 Unicast             : 60 secs
Flapping Delay             : 300 secs
Initial Delay              : 0 secs
Initial Refresh Delay      : 1 secs
Initial Refresh Spread     : 1 secs
Stats Reporting Period     : 0 secs
Queue Route Mon Msg buffer limit : 133693 KB (Current Server Up Count: 2)
Queue Route Mon Msg buffer usage : 0 B
Queue write pulse sent     : Nov 29 19:08:32.826, Nov 29 13:11:53.478 (all)
Queue write pulse received  : Nov 29 19:08:32.826
Update Generation in Progress : No
Reset Walk in Progress     : No
----More-----

```

You can then configure **bmp advertisement-interval** command to set the minimum interval between the sending of BMP routing updates.

Simultaneous Monitoring of Adj-RIB-In Pre-Policy and Post-Policy Views

Table 28: Feature History Table

Feature Name	Release	Description
Simultaneous monitoring of Adj-RIB-In Pre-Policy and Post-Policy views	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8200, 8700); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])</p> <p>You can now monitor BGP events and collect BGP route information both before and after applying policy filters. This feature allows you to view the Adj-RIB-In pre-policy and post-policy views simultaneously for all BGP peers, making it easier to troubleshoot routing policies and verify Remotely Triggered Black Hole (RTBH) routes.</p> <p>This feature introduces these changes:</p> <p>CLI:</p> <p>This feature modifies the following commands:</p> <ul style="list-style-type: none"> • route-monitoring inbound pre-policy • show bgp bmp • YANG Data Model: New XPaths for <ul style="list-style-type: none"> • <code>Cisco-IOS-XR-um-router-bgp-cfg.yang</code> • <code>Cisco-IOS-XR-ipv4-bgp-oper.yang</code> (see GitHub, YANG Data Models Navigator)

This feature allows you to monitor both Adj-RIB-In pre-policy and post-policy views simultaneously for all BGP peers. With this feature, you can monitor BGP events and collect BGP route information both before and after applying policy filters.

With this feature, the following modes are supported:

- **Individual Modes:**

- Inbound pre-policy Route Mirroring



Note Inbound pre-policy Route Mirroring is now supported as per RFC7854.

- Inbound pre-policy Route Monitoring
- Inbound post-policy Route Monitoring
- Local-RIB

- **Combined Modes:**

- Inbound Post-Policy Route-Monitoring and Local-RIB can operate in parallel.
- Inbound Post-Policy and Inbound Pre-Policy Route-Monitoring can operate in parallel.

Benefits of simultaneous monitoring of Adj-RIB-In Pre-Policy and Post-Policy views

These are the benefits of Simultaneous monitoring of Adj-RIB-In Pre-Policy and Post-Policy views feature:

- **Enhanced Monitoring:** Monitors both pre-policy and post-policy routes simultaneously, providing comprehensive visibility into routing changes.
- **High Scalability:** Supports many routes and neighbors, making it suitable for large networks.
- **Improved Debugging:** Facilitates effective debugging of routing policies by providing detailed monitoring information.
- **RTBH Validation:** Ensures that Remote Triggered Black Hole (RTBH) routes are correctly implemented and validated, enhancing network security.

Configure simultaneous monitoring of Adj-RIB-In Pre-Policy and Post-Policy views

Before you begin



Note Before configuring Adj-RIB-In Pre-Policy Route-Monitoring mode, ensure that **soft reconfiguration** is enabled for all BGP Monitoring Protocol (BMP) supported AFIs under all BMP monitored neighbors. **Soft reconfiguration** is a prerequisite for Adj-RIB-In Pre-Policy Route Monitoring.

Procedure

Step 1 Enable BMP on the Router.

Example:

```
Router# configure
Router(config)# bmp server all
```

Step 2 Configure inbound pre-policy mode route monitoring.

Example:

```
Router# configure
Router(config)# bmp server all
Router(config-bgp-bmp)# route-monitoring inbound pre-policy
```

Step 3 Configure inbound post-policy mode route monitoring.

Example:

```
Router# configure
Router(config)# bmp server all
Router(config-bgp-bmp)# route-monitoring inbound post-policy
```

Step 4 Set advertisement interval and scan time.

Example:

```
Router# configure
Router(config)# bmp server all
Router(config-bgp-bmp)# route-monitoring inbound pre-policy
Router(config-bgp-bmp-rmon)# advertisement-interval 15
Router(config-bgp-bmp-rmon)# scan-time 60
```

When both **route-monitoring inbound pre-policy** and **route-monitoring inbound post-policy** are configured at the same time, the system uses the lower values for the advertisement interval and scan time from either the default or configured settings of both modes. For example, if the **advertisement-interval** for pre-policy is set to 30 and for post-policy is set to 15, the value of 15 is used for both modes. This rule also applies to **scan-time**.

Step 5 The default BMP mode is now route-mirroring, where uncompressed BGP updates are sent to the BMP server. You can also explicitly configure this mode using the **route-mirroring inbound pre-policy** command.

Example:

```
Router# configure
Router(config)# bmp server all
Router(config-bgp-bmp)# route-mirroring inbound pre-policy
```

Step 6 Verify that the inbound pre-policy and post-policy views for simultaneous monitoring are enabled.

Example:

```
Router# show bgp bmp server 1 detail
Thu Nov 28 10:53:10.659 UTC
BMP server 1
Host 12::2 Port 17001
NOT Connected
Last Disconnect event received : 00:15:10
Flap count : 1
Precedence: internet
BGP neighbors: 68
```

```

VRF: - (0x60000000)
Update Source: - (-)
Update Source Vrf ID: 0x0
Update Mode : Rmon-In-Pre-Policy, Rmon-In-Post-Policy
  Rmon-In-Pre-Policy
    Advertisement interval : 15 secs
    Advertisement interval (Oper) : 15 secs
    Scanner interval : 60 secs
    Scanner interval (Oper) : 60 secs
  Rmon-In-Post-Policy
    Advertisement interval : 15 secs
    Advertisement interval (Oper) : 15 secs
    Scanner interval : 60 secs
    Scanner interval (Oper) : 60 secs
Flapping Delay : 300 secs
Initial Delay : 0 secs
Initial Refresh Delay : 1 secs
Initial Refresh Spread : 1 secs
Stats Reporting Period : 0 secs
Queue Route Mon Msg buffer limit : 503316 KB (Current Server Up Count: 0)
Queue Route Mon Msg buffer usage : 0 B
Queue write pulse sent : Nov 28 10:33:23.988, Nov 28 10:38:00.258 (all)
Queue write pulse received : Nov 28 10:38:00.258
Update Generation in Progress : No
Reset Walk in Progress : No
IPv4 Unicast
  Version : 416
  Rmon-In-Pre-Policy:
    Init EOR Version : 416
    Init EOR Pending count : 0
  Rmon-In-Post-Policy:
    Init EOR Version : 416
    Init EOR Pending count : 0
  Update Generation
    Last Run : Nov 28 10:33:23.982, Count 1
    Walk Currently Stalled : No, Last Stalled : not set, Count 0
VPNv4 Unicast
  Version : 200
  Rmon-In-Pre-Policy:
    Init EOR Version : 200
    Init EOR Pending count : 0
  Rmon-In-Post-Policy:
    Init EOR Version : 200
    Init EOR Pending count : 0
  Update Generation
    Last Run : Nov 28 10:33:23.984, Count 1
    Walk Currently Stalled : No, Last Stalled : not set, Count 0
IPv6 Unicast
  Version : 505
  Rmon-In-Pre-Policy:
    Init EOR Version : 505
    Init EOR Pending count : 0
  Rmon-In-Post-Policy:
    Init EOR Version : 505
    Init EOR Pending count : 0
  Update Generation
    Last Run : Nov 28 10:33:23.985, Count 1
    Walk Currently Stalled : No, Last Stalled : not set, Count 0
VPNv6 Unicast
  Version : 320
  Rmon-In-Pre-Policy:
    Init EOR Version : 320
    Init EOR Pending count : 0
  Rmon-In-Post-Policy:

```

```
Init EOR Version      : 320
Init EOR Pending count : 0
Update Generation
  Last Run            : Nov 28 10:33:23.987, Count 1
  Walk Currently Stalled : No, Last Stalled : not set, Count 0

TCP:
  Last message sent: Nov 28 10:33:24.215, Status: Not Connected
  Last write pulse received: Nov 28 10:33:24.311, Waiting: FALSE

Message Stats:
Total msgs dropped   : 0
Total msgs pending   : 0, Max: 587 at Nov 28 10:33:23.988
Total messages sent   : 712
Total bytes sent      : 133462, Time spent: 0.006 secs
  INITIATION          : 1
  TERMINATION          : 0
  STATS-REPORT         : 0
  PER-PEER messages    : 711

  PEER UP messages     : 68
  PEER DOWN messages   : 0
  ROUTE-MIR messages   : 0
  ROUTE-MON messages   : 643
  Last RMon EnQ        : Nov 28 10:33:23.988
  EOR messages         : 278
  Update messages      : 365 (Prefixes: 3254, Err: 0)
Update gen time spent: 0.005 secs

Neighbor 2001:DB8:1:5 (vrf vrf1)
Last BMP Activate : Nov 28 10:33:15.065
First Peer Up EnQ : Nov 28 10:33:15.065
Last Peer Up EnQ  : Nov 28 10:33:15.065
Last EOR Pending  : not set
Last EOR EnQ      : not set
Last RMon EnQ     : Nov 28 10:33:23.988
Messages pending  : 0
Messages dropped  : 0
Messages sent     : 3
  PEER-UP         : 1
  PEER-DOWN       : 0

Rmon-In-Pre-Policy:
ROUTE-MON         : 1
Update            : 1 (Prefixes: 10, Err: 0)

Rmon-In-Post-Policy:
ROUTE-MON         : 1
Update            : 1 (Prefixes: 10, Err: 0)

Neighbor 2001:DB8:3:5 (vrf vrf1)
Last BMP Activate : Nov 28 10:33:15.065
First Peer Up EnQ : Nov 28 10:33:15.065
Last Peer Up EnQ  : Nov 28 10:33:15.065
Last EOR Pending  : not set
Last EOR EnQ      : not set
Last RMon EnQ     : Nov 28 10:33:23.988
Messages pending  : 0
Messages dropped  : 0
Messages sent     : 3
  PEER-UP         : 1
  PEER-DOWN       : 0

Rmon-In-Pre-Policy:
```

```

ROUTE-MON      : 1
Update        : 1 (Prefixes: 10, Err: 0)

Rmon-In-Post-Policy:
ROUTE-MON      : 1
Update        : 1 (Prefixes: 10, Err: 0)

Neighbor 2001:DB8:2:5 (vrf vrf1)
Last BMP Activate : Nov 28 10:33:15.065
First Peer Up EnQ : Nov 28 10:33:15.065
Last Peer Up EnQ  : Nov 28 10:33:15.065
Last EOR Pending  : not set
Last EOR EnQ      : not set
Last RMon EnQ     : Nov 28 10:33:23.988
Messages pending  : 0
Messages dropped  : 0
Messages sent     : 3
PEER-UP          : 1
PEER-DOWN        : 0

```

....

Step 7

To view the information about prefixes, advertisement status, and paths learned from the monitored BGP neighbors, use the **show bgp bmp mode route-monitoring inbound post-policy table** command.

Example:

```

Router# show bgp bmp mode route-monitoring inbound post-policy table
Thu Nov 28 10:54:35.570 UTC

```

```

-----
Route monitoring mode : Inbound Post-Policy
-----

```

Bitmap and Status Codes:

```

u: Update send pending      U: Update sent
w: Withdraw send pending    W: Withdraw sent, path awaiting deletion
D: Server down              .: Server not configured or mode not applicable

```

```

Address-Family : IPv4 Unicast

```

```

-----
BMP prefix table version : 1196

```

Network	Version	Neighbor	Adv Status	Flags	Path ID
			87654321		
10.2.2.2/8	1	10.2.2.2DD	0x0e	1
10.2.0.0/8	2	10.2.2.2DD	0x0e	1
10.0.1.0/8	3	10.2.2.2DD	0x0e	1
10.0.2.0/8	4	10.2.2.2DD	0x0e	1
10.0.0.0/8	5	10.2.2.2DD	0x0e	1
192.168.0.1/16	6	10.2.2.2DD	0x0e	1
209.165.201.1/27	7	10.2.2.2DD	0x0e	4
209.165.201.1/27	7	10.2.2.2DD	0x0e	3
209.165.201.1/27	7	10.2.2.2DD	0x0e	2
209.165.201.1/27	7	10.2.2.2DD	0x0e	1
209.165.201.1/27	7	10.2.2.2DD	0x0e	7
209.165.201.1/27	7	10.2.2.2DD	0x0e	6
209.165.201.1/27	7	10.2.2.2DD	0x0e	5
209.165.201.1/27	7	10.2.2.2DD	0x0e	10
209.165.201.1/27	7	10.2.2.2DD	0x0e	8
209.165.201.10/27	8	10.2.2.2DD	0x0e	4
209.165.201.10/27	8	10.2.2.2DD	0x0e	3

....

- a) To view the pre-policy and post-policy advertisement statuses in the same table, use the **show bgp bmp mode route-monitoring inbound all table** command.

Example:

```
Router# show bgp bmp mode route-monitoring inbound all table
Thu Nov 28 10:55:03.096 UTC
```

```
-----
Route monitoring mode : Inbound All
-----
```

Bitmap and Status Codes:

```
u: Update send pending      U: Update sent
w: Withdraw send pending    W: Withdraw sent, path awaiting deletion
D: Server down              .: Server not configured or mode not applicable
```

```
Address-Family : IPv4 Unicast
-----
```

```
BMP prefix table version : 1196
```

Network	Version	Neighbor	Adv Status (Pre)	Adv Status (Post)	Flags	Path ID
			87654321	87654321		
10.2.2.2/8	1	10.2.2.2DDDD	0x0e	1
10.2.0.0/8	2	10.2.2.2DDDD	0x0e	1
10.0.1.0/8	3	10.2.2.2DDDD	0x0e	1
10.0.2.0/8	4	10.2.2.2DDDD	0x0e	1
10.0.0.0/8	5	10.2.2.2DDDD	0x0e	1
192.168.0.1/16	6	10.2.2.2DDDD	0x0e	1
209.165.201.1/27	7	10.2.2.2DDDD	0x0e	4
209.165.201.1/27	7	10.2.2.2DDDD	0x0e	3
209.165.201.1/27	7	10.2.2.2DDDD	0x0e	2
209.165.201.1/27	7	10.2.2.2DDDD	0x0e	1
209.165.201.1/27	7	10.2.2.2DDDD	0x0e	7
209.165.201.1/27	7	10.2.2.2DDDD	0x0e	6
209.165.201.1/27	7	10.2.2.2DDDD	0x0e	5
209.165.201.1/27	7	10.2.2.2DDDD	0x0e	10
209.165.201.1/27	7	10.2.2.2DDDD	0x0e	9
209.165.201.1/27	7	10.2.2.2DDDD	0x0e	8
209.165.201.11/27	8	10.2.2.2DDDD	0x0e	4
209.165.201.11/27	8	10.2.2.2DDDD	0x0e	3
....						

This sample output provides the advertisement statuses of both Pre-Policy and Post-Policy views.

BGP—Multiple Cluster IDs

The BGP—Multiple Cluster IDs feature allows an iBGP neighbor (usually a route reflector) to have multiple cluster IDs: a global cluster ID and additional cluster IDs that are assigned to clients (neighbors). Prior to the introduction of this feature, a device could have a single, global cluster ID.

When a network administrator configures per-neighbor cluster IDs:

- The loop prevention mechanism based on a CLUSTER_LIST is automatically modified to take into account multiple cluster IDs.
- A network administrator can disable client-to-client route reflection based on cluster ID.

Restriction

The BGP Multiple Cluster-IDs feature only works in default VRF.

BGP Flowspec Overview

Table 29: Feature History Table

Feature Name	Release Information	Feature Description
BGP Flowspec	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700) (select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*)</p> <p>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.</p> <p>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.</p> <p>*This functionality is now supported on:</p> <ul style="list-style-type: none"> • 8712-MOD-M • 88-LC1-12TH24FH-E

Table 30: Feature History Table

Feature Name	Release Information	Feature Description
Scaling BGP Flowspec to 6000 Rules	Release 7.5.2	<p>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.</p> <p>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).</p>

The BGP flow specification (flowspec) feature allows you to rapidly deploy and propagate filtering and policing functionality among many BGP peer routers to mitigate the effects of a distributed denial-of-service (DDoS) attack over your network.

BGP Flowspec feature allows you to construct instructions to match a particular flow with IPv4 and IPv6 source, IPv4 and IPv6 destination, L4 parameters and packet specifics such as length, fragment, destination

port and source port, actions that must be taken, such as dropping the traffic, or policing it at a definite rate, or redirect the traffic, through a BGP update. In the BGP update, the flowspec matching criteria is represented by Network Layer Reachability Information (BGP NLRI) and the actions are represented by BGP extended communities.

You can use the BGP Flowspec feature for mitigation of DDoS attack. When a DDoS attack occurs on a particular host inside a network, you can send a flowspec update to the border routers so that the attack traffic can be policed or dropped, or even redirected elsewhere. For example, to an appliance that cleans the traffic by filtering out the bad traffic and forward only the good traffic toward the affected host.

Once flowspecs have been received by a router and programmed in applicable line cards, any active L3 ports on those line cards start processing ingress traffic according to flowspec rules.

The BGP Flowspec feature cannot coexist with MAP-E and PBR on a given interface. If you configure BGP Flowspec with PBR, the router does not display any error or system message. The router ignores the BGP Flowspec configuration and the feature will not function.

Flow Specifications

A flow specification is an n-tuple consisting of several matching criteria that can be applied to IP traffic. A given IP packet matches the defined flow if it matches all the specified criteria.

Every flow-spec route is effectively a rule, consisting of a matching part (encoded in the NLRI field) and an action part (encoded as a BGP extended community). The BGP flowspec rules are converted internally to equivalent C3PL policy representing match and action parameters. The match and action support can vary based on underlying platform hardware capabilities. Sections *Supported Matching Criteria and Actions* and *Traffic Filtering Actions* provide information on the supported match (tuple definitions) and action parameters.



Note

- Cisco 8800 series routers support up to 6,000 flowspec rules.
- Cisco 8200 and 8100 series routers support up to 3,000 flowspec rules.

Supported Matching Criteria and Actions

Table 31: Feature History Table

Feature Name	Release Name	Description
Additional BGP FlowSpec Actions for Enhanced Security	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *Additional BGP FlowSpec Actions for Enhanced Security is now supported on Cisco 8712-MOD-M routers.

Feature Name	Release Name	Description
Additional BGP FlowSpec Actions for Enhanced Security	Release 7.3.3	<p>This release introduces additional BGP FlowSpec actions for enhanced security against distributed denial-of-service (DDoS) attacks.</p> <ul style="list-style-type: none"> • Redirect Nexthop VRF only: Redirects the traffic to a different Autonomous System Number (ASN). • Rate Limit and Redirect IPv4 or IPv6 Nexthop: Redirects the traffic to the indicated nexthop IPv4 or IPv6 address. Policer rate regulates the traffic. • Rate Limit and Redirect Nexthop VRF: Redirects the traffic to the next hop IPv4 address through a VRF. Policer rate regulates the traffic. This action is supported only on Q200 Silicon One ASIC.

Table 32: Feature History Table

Feature Name	Release Name	Description
BGP FlowSpec NLRI types	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700) (select variants only*)</p> <p>*BGP FlowSpec NLRI types is now supported on Cisco 8712-MOD-M routers.</p>

Feature Name	Release Name	Description
BGP FlowSpec NLRI types	Release 7.3.15	<p>A BGP flow specification consists of several matching criteria encoded in the NLRI that is applied to IP traffic. A given IP packet must match all the specified criteria. Network layer reachability information (NLRI) exchanges routing information and matching criteria between BGP peers, indicating how to reach the destination.</p> <p>The following NLRI types are supported:</p> <ul style="list-style-type: none"> • Type 7: IPv4 or IPv6 ICMP type • Type 8: IPv4 or IPv6 ICMP code • Type 9: IPv4 TCP flags (2 bytes include reserved bits) • Type 10: IPv4 Packet length • Type 11: IPv4 or IPv6 DSCP • Type 12: IPv4 fragmentation bits
BGP FlowSpec Actions	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700) (select variants only*)</p> <p>*BGP FlowSpec Actions is now supported on Cisco 8712-MOD-M routers.</p>
BGP FlowSpec Actions	Release 7.3.15	<p>This feature provides information on the actions that can be associated with a BGP flow. The traffic filtering flow specification is applied based on the specified rule. The following extended community values that can be used to specify particular action:</p> <ul style="list-style-type: none"> • Set DSCP • Redirect IPv4 or IPv6 next hop

Overview

A flow specification NLRI type may include several components such as destination prefix, source prefix, protocol, ports, and so on. This NLRI is treated as an opaque bit string prefix by BGP. Each bit string identifies a key to a database entry with which a set of attributes can be associated. This NLRI information is encoded using MP_REACH_NLRI and MP_UNREACH_NLRI attributes. Whenever the corresponding application does not require Next-Hop information, this is encoded as a 0-octet length Next Hop in the MP_REACH_NLRI attribute, and ignored. The NLRI field of the MP_REACH_NLRI and MP_UNREACH_NLRI is encoded as a 1- or 2-octet NLRI length field followed by a variable-length NLRI value. The NLRI length is expressed in octets.

The flow specification NLRI type consists of several optional sub-components. A specific packet is considered to match the flow specification when it matches the intersection of all the components present in the specification. The following are the supported component types or tuples that you can define:

BGP Flowspec NLRI type	QoS Match Fields	Description and Syntax Construction	Value Input Method
Type 1	IPv4 or IPv6 destination address	<p>Defines the destination prefix to match. Prefixes are encoded in the BGP UPDATE messages as a length in bits followed by enough octets to contain the prefix information.</p> <p>Encoding: <type (1 octet), prefix length (1 octet), prefix></p> <p>Syntax:</p> <p>match destination-address {ipv4 ipv6} <i>address/mask length</i></p>	Prefix length
Type 2	IPv4 or IPv6 source address	<p>Defines the source prefix to match.</p> <p>Encoding: <type (1 octet), prefix-length (1 octet), prefix></p> <p>Syntax:</p> <p>match source-address {ipv4 ipv6} <i>address/mask length</i></p>	Prefix length
Type 3	IPv4 or IPv6 protocol	<p>Contains a set of {operator, value} pairs that are used to match the IP protocol value byte in IP packets.</p> <p>Encoding: <type (1 octet), [op, value]+></p> <p>Syntax:</p> <p>match protocol {<i>protocol-value</i> [<i>min-value</i> – <i>max-value</i>]}</p>	<p>Single value</p> <p>Note Multi-value range is not supported</p>

Type 4	IPv4 or IPv6 source or destination port	<p>Defines a list of {operation, value} pairs that matches source or destination TCP or UDP ports. Values are encoded as 1- or 2-byte quantities. Port, source port, and destination port components evaluate to FALSE if the IP protocol field of the packet has a value other than TCP or UDP. If the packet is fragmented and this is not the first fragment, or if the system is unable to locate the transport header.</p> <p>Encoding: <type (1 octet), [op, value]+></p> <p>Syntax:</p> <p>match source-port { <i>source-port-value</i> <i>min-value</i> - <i>max-value</i> }</p> <p>match destination-port { <i>destination-port-value</i> <i>min-value</i> - <i>max-value</i> }</p>	Multi-value range
Type 5	IPv4 or IPv6 destination port	<p>Defines a list of {operation, value} pairs used to match the destination port of a TCP or UDP packet. Values are encoded as 1- or 2-byte quantities.</p> <p>Encoding: <type (1 octet), [op, value]+></p> <p>Syntax:</p> <p>match destination-port { <i>destination-port-value</i> [<i>min-value</i> - <i>max-value</i>] }</p>	Multi-value range
Type 6	IPv4 or IPv6 Source port	<p>Defines a list of {operation, value} pairs used to match the source port of a TCP or UDP packet. Values are encoded as 1- or 2-byte quantities.</p> <p>Encoding: <type (1 octet), [op, value]+></p> <p>Syntax:</p> <p>match source-port { <i>source-port-value</i> [<i>min-value</i> - <i>max-value</i>] }</p>	Multi-value range

Type 7	IPv4 or IPv6 ICMP type	<p>Defines a list of {operation, value} pairs used to match the type field of an ICMP packet. Values are encoded using a single byte. The ICMP type and code specifiers evaluate to FALSE whenever the protocol value is not ICMP.</p> <p>Encoding: <type (1 octet), [op, value]+></p> <p>Syntax:</p> <p>match {ipv4 ipv6} icmp-type {value min-value -max-value}</p>	<p>Single value</p> <p>Note Multi-value range is not supported</p>
Type 8	IPv4 or IPv6 ICMP code	<p>Defines a list of {operation, value} pairs used to match the code field of an ICMP packet. Values are encoded using a single byte.</p> <p>Syntax:</p> <p>Encoding: <type (1 octet), [op, value]+></p> <p>match {ipv4 ipv6} icmp-type {value min-value -max-value}</p>	<p>Single value</p> <p>Note Multi-value range is not supported</p>
Type 9	<p>IPv4 or IPv6 TCP flags (2 bytes include reserved bits)</p> <p>Note Reserved and NS bit not supported</p>	<p>Bitmask values can be encoded as a 1- or 2-byte bitmask. When a single byte is specified, it matches byte 13 of the TCP header, which contains bits 8 through 15 of the 4th 32-bit word. When a 2-byte encoding is used, it matches bytes 12 and 13 of the TCP header with the data offset field having a "don't care" value. As with port specifier, this component evaluates to FALSE for packets that are not TCP packets. This type uses the bitmask operand format, which differs from the numeric operator format in the lower nibble.</p> <p>Encoding: <type (1 octet), [op, bitmask]+></p> <p>Syntax:</p> <p>match tcp-flag value bit-mask mask_value</p>	<p>Bit mask</p>

Type 10	<p>IPv4 or IPv6 Packet length</p> <p>Starting from Release 7.10.1, the IPv6 packet length is supported.</p> <p>Note</p> <ul style="list-style-type: none"> Reserved and NS bit not supported IPv4 or IPv6 support is available for the packets that are not the first fragment packets. 	<p>Match on the total IP packet length (excluding Layer 2, but including IP header). Values are encoded using 1- or 2-byte quantities.</p> <p>Encoding: <type (1 octet), [op, value]+></p> <p>Syntax:</p> <p>matchpacket length {packet-length-value min-value -max-value}</p>	Multi-value range
Type 11	IPv4 or IPv6 DSCP	<p>Defines a list of (operation, value) pairs used to match the 6-bit DSCP field. Values are encoded using a single byte, whereas the two most significant bits are zero and the six least significant bits contain the DSCP value.</p> <p>Note</p> <p>The DSCP does not contain Flowspec statistics.</p> <p>Encoding: <type (1 octet), [op, value]+></p> <p>Syntax:</p> <p>match dscp {dscp-value min-value -max-value}</p>	Multi-value range
Type 12	<p>IPv4 Fragmentation bits</p> <p>Note</p> <p>IPv4 support is available for the packets that are not the first fragment packets.</p> <p>IPv6 BGP flowspec does not supports Type 12 NRLI.</p>	<p>Identifies a fragment-type as the match criterion for a class map.</p> <p>Encoding: <type (1 octet), [op, bitmask]+></p> <p>Syntax:</p> <p>match fragment type [is-fragment]</p>	Bit mask

In a given flowspec rule, 2-tuple action combinations can be specified without restrictions. However, mixing address family between matching criterion and actions are not allowed. For example, IPv4 matches cannot be combined with IPv6 actions and vice versa.

Limitations for BGP FlowSpec

These limitations apply to the BGP FlowSpec feature.

- BGP Flowspec statistics are supported when there is a policer rate limit.
The policer action scale is limited to a maximum of 128 per slice.
- BGP Flowspec statistics are supported in Redirect action only when a policer is attached. BGP Flowspec statistics is not supported for Redirect action alone.
- VRF to default VRF redirect is not supported.

BGP flowspec redirect from global VRF to L3VPN and segment routing policy

Table 33: Feature History Table

Feature Name	Release Information	Feature Description
BGP flowspec redirect from global VRF to L3VPN and segment routing policy	Release 24.4.1	<p>Introduced in this release on: Fixed Systems(8200, 8700)(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*).</p> <p>This feature that enables 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 for an enhanced network routing efficiency is now supported on the following hardware.</p> <p>*This feature is now supported on:</p> <ul style="list-style-type: none"> • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 88-LC1-36EH
BGP flowspec redirect from global VRF to L3VPN and segment routing policy	Release 24.2.11	<p>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.</p>

The BGP Flowspec Redirect from Global VRF to L3VPN and Segment Routing policy feature from Global VRF to L3VPN feature allows traffic to be dynamically redirected 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.

BGP Flowspec Topology

Figure 13: Forwarding based on SR-Policy

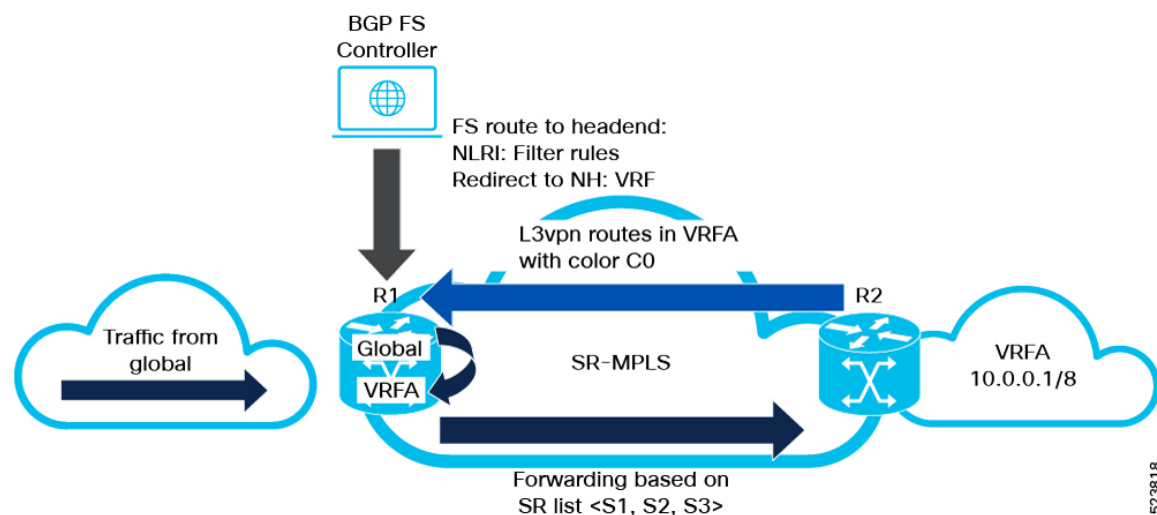
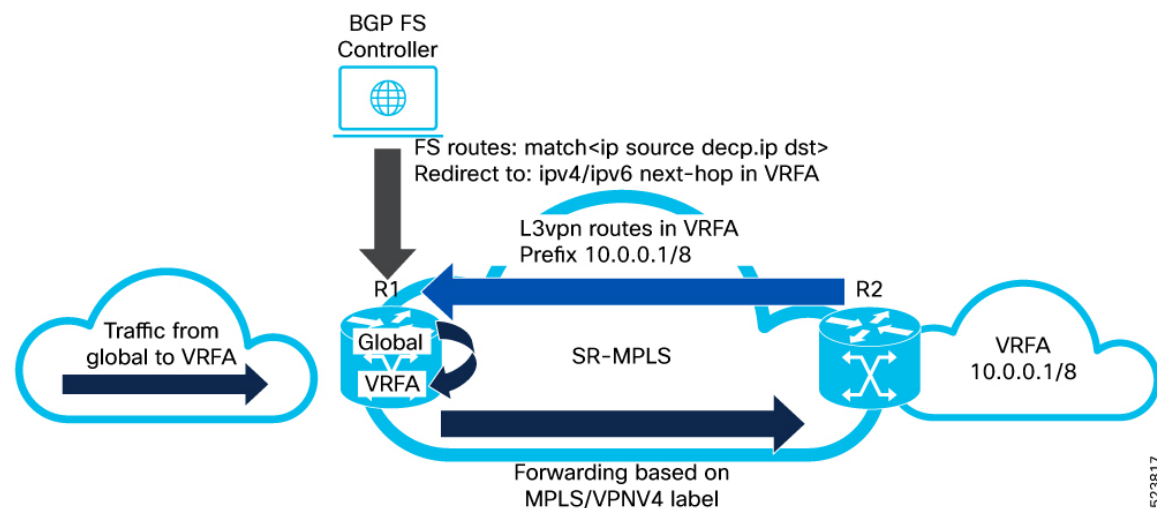


Figure 14: Forwarding based on MPLS



A network traffic arrives from an interface VRFA. However, this interface is not specifically designated for customers. The incoming packet has a destination IP address of 10.0.0.1/8 and this IP address is not available in the global routing table—it exists only in the VRF routing table. As a result, the packet is dropped.

To address this issue, we apply certain criteria. We ensure that the IP address lookup for such packets occurs within the customer VRF rather than the global VRF. By doing so, we direct the packet to the correct routing context, allowing successful forwarding.

The forwarding process in this scenario typically happens through L3VPN or SR-Policy, which provide an effective mechanism for managing routing and forwarding in complex network environments. For those kind of route we match certain criteria and make sure the lookup for the IP address happens in the customer VRF and not the global VRF.

The BGP flowspec server is where the rule is initially programmed. These rules are then propagated to the BGP flowspec neighbor through This should be BGP Network Layer Reachability Information (NLRI). Once the client receives the rule, it is stored in the database. When the rule becomes active, it starts taking effect. Incoming packets are matched against this active rule.

If a packet meets the criteria specified by the rule, it is redirected to the appropriate VRF instance. In the redirect action, it is crucial to specify the correct route target. This ensures that the packet is correctly routed to the intended VRF.

The BGP flowspec server plays a pivotal role in defining and enforcing traffic rules, allowing for fine-grained control over packet handling within the network.

Configure BGP Flowspec Redirect from Global VRF

Configuration Examples

Perform the steps given below on the BGP Flowspec controller to enable BGP Flowspec redirect from global VRF.

- Create a Class Map - Create a class map to be used for matching packets to the class whose name you specify and enters the class map configuration mode.
- Build a Policy Map - Create a policy map that can be attached to a flowspec to specify a service policy and enters the policy map configuration mode.
- Link the Class Map to the Policy Map - The policy map configuration mode, the 'class type traffic' command is used to associate the previously configured traffic class with the policy map.
- Define Policy actions- Defines the actions that you want to perform.

```
/* Create a Class Map */
Router# config
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)# exit
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

/* Build a Policy Map */
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)# exit

/* Link the Class Map to the 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-c)# end-policy-map
Router(config)# exit

/* Define Policy actions */
Router(config)# flowspec
Router(config)# address-family ipv4
Router(config-af)# service-policy type pbr ipv4_PM1
Router(config)# flowspec
Router(config)# address-family ipv6
Router(config-af)# service-policy type pbr ipv6_PM1

```

For information on how to configure SR-Policy refer the chapter "*Configure SRv6 Traffic Engineering*" in the "*Segment Routing Configuration Guide for Cisco 8000 Series Routers*".

Similarly, for information on how to configure L3VPN refer the chapter "*Implementing MPLS Layer 3 VPNs*" in the "*L3VPN Configuration Guide for Cisco 8000 Series Routers*".

Running Configuration

```

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
```

Verification



Note BGP Flowspec statistics will not be available until a policer action is configured.

Verify the number of BGP Flowspec entries present in the OFA object.

```
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
```

Verify the BGP Flowspec rules and statistics.

```
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
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

The default action for a traffic filtering flow specification is to accept IP traffic that matches that particular rule. The following extended community values can be used to specify particular actions:



Note The BGP flowspec actions *rate limit* and *redirect* are not supported together.

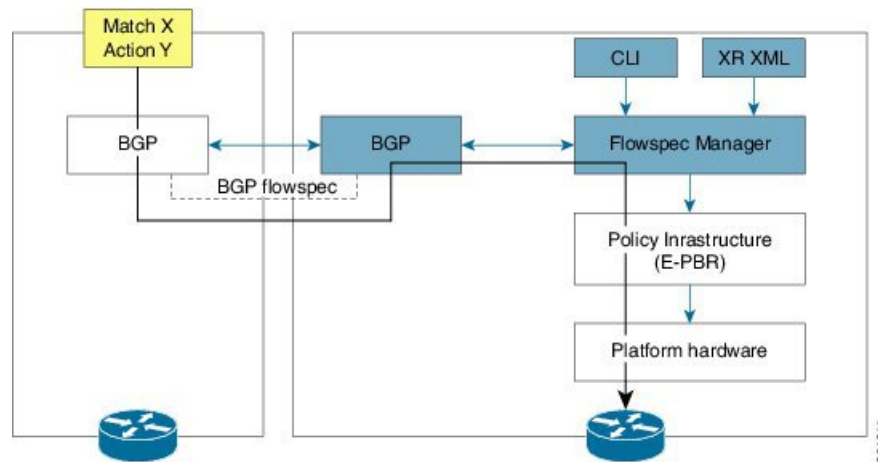
The BGP flowspec action *redirect* is supported only for nexthop IPv4 and IPv6 not with nexthop VRF IPv4 and IPv6.

Type	Extended Community	PBR Action	Description

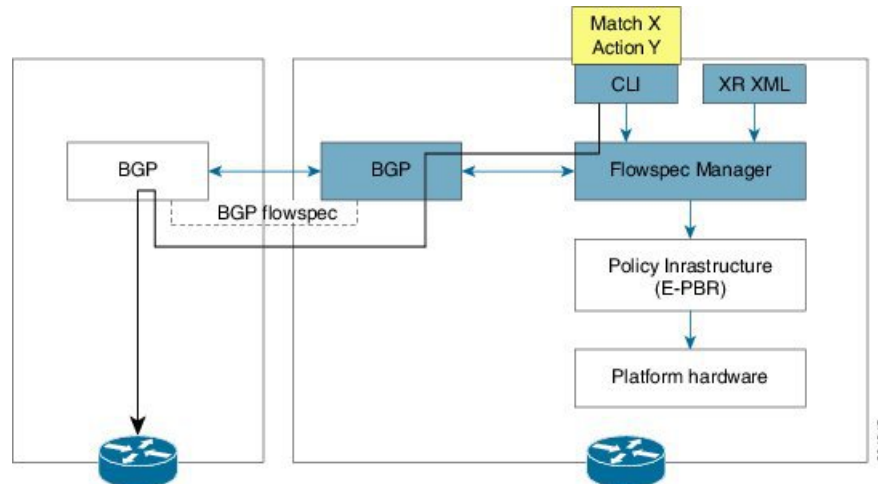
0x8006	traffic-rate 0 traffic-rate <rate>	Drop Police	<p>The traffic-rate extended community is a non-transitive extended community across the autonomous-system boundary and uses following extended community encoding:</p> <p>The first two octets carry the 2-octet id, which can be assigned from a 2-byte 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.</p> <p>Command syntax</p> <p>police rate < > drop</p>
0x8009	traffic-marking	Set DSCP	<p>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.</p> <p>Command syntax</p> <p>set dscp <6 bit value></p>
0x0800	Redirect IP NH	Redirect IPv4 or IPv6 Nexthop	<p>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.</p> <p>Note</p> <p>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.</p> <p>Note</p> <p>Redirect IP NH is supported only in default VRF.</p> <p>Command syntax</p> <p>redirect {ipv4 ipv6} next-hop {ipv4-address ipv6-address}</p>

BGP Flowspec Client-Server Controller Model

The BGP Flowspec model comprises of a client and a server Controller. The Controller is responsible for sending or injecting the flowspec NLRI entry. The client (acting as a BGP speaker) receives that NLRI and programs the hardware forwarding to act on the instruction from the Controller. An illustration of this model is provided below.

BGP Flowspec Client

Here, the Controller on the left-hand side injects the flowspec NRLI, and the client on the right-hand side receives the information, sends it to the flowspec manager, configures the ePBR (Enhanced Policy-based Routing) infrastructure, which in turn programs the hardware from the underlying platform in use.

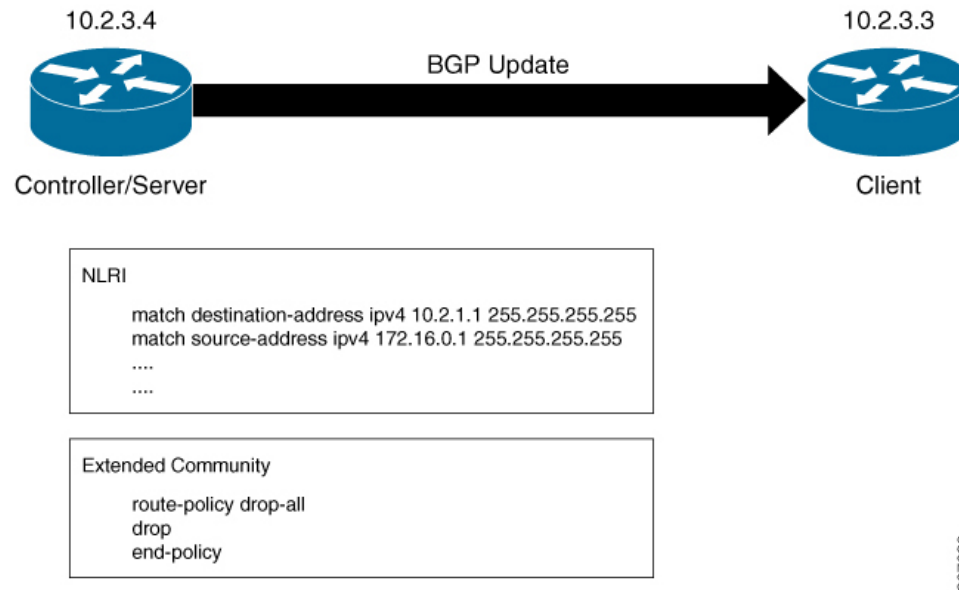
BGP Flowspec Controller

The Controller is configured using CLI to provide an entry for NRLI injection.

Configure BGP Flowspec

The following sections show how to configure BGP Flowspec feature.

Figure 15: BGP Flowspec



The controller or the server with IP address 10.2.3.4 sends the Flowspec NLRI to the client with IP address 10.2.3.3. The NLRI consists of matching criteria, the client processes based on this criteria. Traffic is dropped or accepted based on the configured criteria.

The following section describes how you can configure BGP Flowspec on the client:

```

/* Define a Virtual Routing and Forwarding (VRF) instance named vrfl and set up
import and export route targets for different address families. */

```

```

Router(config)# router bgp 140
Router(config-bgp)# vrf vrfl
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

/* Configure BGP Flowspec both on the server and client side. */
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

/* Configure the policy to accept all presented routes without modifying the routes */
Router(config) # route-policy pass-all
Router(config) # pass
Router(config) # end-policy

/* Configure the policy to reject all presented routes without modifying the routes */
Router(config) # route-policy drop-all
Router(config) # drop
Router(config) # end-policy

/* Configure BGP towards flowspec server */
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

/* Disable BGP Flowspec */
Router(config)# interface bundle-ether 3.1
Router(config-subif)# ipv4 flowspec disable
Router(config-subif)# 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 */
Router(config)# route-policy pass-all
Router(config)# pass
Router(config)# end-policy

/* Configure the policy to reject all presented routes without modifying the routes */
Router(config)# route-policy drop-all
Router(config)# drop
Router(config)# end-policy

/* Configure BGP towards flowspec client */
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

/* Configure IPv4 flowspec to be advertised to client. Define traffic classes. */
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

/* Define a policy map and associate it with traffic classes. */
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
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

/* Configure IPv6 flowspec to be advertised to client. Define traffic classes. */
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

/* Define a policy map and associate it with traffic classes.
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

/* Class map configuration with DSCP */
Router(config-map)# 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

/* Policy map configuration with IPv4 Redirect and Rate Limiter */
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

```

Running Configuration

```

/* Define a Virtual Routing and Forwarding (VRF) instance named vrfl and set up
import and export route targets for different address families. */
router bgp 140
vrf vrfl
  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

```

Verification

The following show output displays the status of the flowspec from the client side.

```

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 ?

Router# show bgp ipv6 flowspec

BGP 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: 1503
BGP main routing table version 1504
BGP NSR Initial initsync version 2 (Reached)
BGP NSR/ISSU Sync-Group versions 1504/0
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: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

Router# show bgp vpnv4 flowspec
BGP 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: 0
BGP main routing table version 5

```

```

BGP NSR Initial initsync version 3 (Reached)
BGP NSR/ISSU Sync-Group versions 5/0
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
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

Router# show bgp vpnv6 flowspec
BGP 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: 0
BGP main routing table version 5
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 5/0
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
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

Router# show bgp ipv6 flowspec summary
BGP 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: 1503
BGP main routing table version 1504
BGP NSR Initial initsync version 2 (Reached)
BGP NSR/ISSU Sync-Group versions 1504/0
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.
Process RcvTblVer bRIB/RIB LabelVer ImportVer SendTblVer StandbyVer
Speaker 1504 1504 1504 1504 1504 1504
Neighbor Spk AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down St/PfxRcd
200.255.1.5 0 4787 6957 2957 1504 0 0 04:48:02 0
200.255.1.6 0 50011 3015 3010 0 0 0 05:27:50 (NoNeg)
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

Router# show bgp vpnv4 flowspec summary
BGP 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: 0
BGP main routing table version 5
BGP NSR Initial initsync version 3 (Reached)
BGP NSR/ISSU Sync-Group versions 5/0
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.
Process RcvTblVer bRIB/RIB LabelVer ImportVer SendTblVer StandbyVer
Speaker 5 5 5 5 5
Neighbor Spk AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down St/PfxRcd
202.158.2.1 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

```

```

Router# show bgp vpnv6 flowspec summary
BGP 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: 0
BGP main routing table version 5
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 5/0
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.
Process RcvTblVer bRIB/RIB LabelVer ImportVer SendTblVer StandbyVer
Speaker 5 5 5 5 5
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

```

```

Router# show flowspec ipv4 detail
AFI: IPv4
Flow :Dest:10.1.1.1/32,Proto:=6,DPort:=80,SPort:=3000,Length:=200,DSCP:=10
Actions :Traffic-rate: 0 bps (bgp.1)
Statistics (packets/bytes)
Matched : 18174999/3707699796
Transmitted : 0/0
Dropped : 18174999/3707699796

```

```

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

```

```

Router# 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

```

```

Router# show flowspec vrf customer_1 ipv6 detail
VRF: customer_1 AFI: IPv6

```

```
Router# show flowspec ipv4 detail
Flow          :Dest:192.0.2.254/24,DSCP:>=10&amp;&lt;=12
Actions       :Traffic-rate: 5000000 bps Nexthop: 10.26.245.2   (bgp.1)
Statistics    (packets/bytes)
  Matched     : 1169087/233817400
  Transmitted : 369952/73990400
  Dropped     : 799135/159827000
```

Enabling BGP Flowspec for IPv6 Packet Length

Table 34: Feature History Table

Feature Name	Release Information	Feature Description
Enabling BGP Flowspec for IPv6 Packet Length	Release 7.10.1	<p>Services such as end-to-end security, quality of service (QoS), and globally unique addresses are now supported for IPv6 packet lengths, which allows your networks to scale and provides them with global reachability. Support for IPv6 packet lengths also means that, in terms of the matching criteria, support for BGP Network Layer Reachability Information (BGP NLRI) type-10 flowspec for IPv6 is added.</p> <p>This feature introduces the following to enable BGP flowspec for IPv6 packet length:</p> <ul style="list-style-type: none"> • CLI: Introduces the hw-module profile flowspec ipv6-packet-len-enable command. • YANG Data Model: New XPaths for <code>Cisco-IOS-XR-um-8000-hw-module-profile-cfg.yang</code> (see GitHub, YANG Data Models Navigator).

An IPv6 address has 128 bits, or 16 bytes. The address is divided into eight 16-bit hexadecimal blocks separated by colons (:) in the format: x:x:x:x:x:x:x:x. BGP Flowspec match conditions for IPv6 packet length support the standard length of 16 bits (2 bytes) or /128 IPv6 source IP address matches. By default, this IPv6 packet length is disabled.

This feature introduces the **hw-module profile flowspec ipv6-packet-len-enable** command that enables BGP Flowspec for IPv6 packet length. Support for IPv6 packet lengths also means that, in terms of the matching criteria, support for BGP Network Layer Reachability Information (BGP NLRI) type-10 flowspec for IPv6 is added.

See [Supported Matching Criteria](#) for details on BGP NLRI Flowspec types and their matching fields.



Note After configuring the command, you must reload the router for the feature to take effect.

Restriction

- This packet length feature is supported only in the ingress direction for non-compression ACLs.
- This feature is supported on:
 - 8201-32FH
 - 88-LC0-36FH-M
 - 88-LC0-36FH-MO
 - 8102-64H

- 8101-32H
- 8101-32H-O
- 8101-32FH
- 8202-32FH-M
- 88-LC0-34H14FH
- 88-LC1-36EH

Configuration

To enable BGP flowspec IPv6 packet length, perform the following actions:

1. Enter the IOS XR configuration mode.

```
Router#config
```

2. Enable the flowspec IPv6 packet length profile for an IPv6 interface.

```
Router(config)#hw-module profile flowspec ipv6-packet-len-enable
```

```
Thu Dec 15 09:15:49.226 UTC
```

In order to activate/deactivate this flowspec IPv6 packet-len profile, you must manually reload the chassis/all line cards

3. Commit the changes.

```
Router(config)#commit
```

After configuring the command, you must reload the router for the feature to take effect.

You can then configure IPv6 flowspec on the server router which acts as a BGP flowspec (bgpfs) server, and then define a policy map and associate it with traffic classes.

```
Router(config)# class-map type traffic match-all class1
```

```
Router(config-cmap)# match protocol tcp
```

```
Router(config-cmap)# match destination-address ipv6 2:1:1::1/64
```

```
Router(config-cmap)# match packet length 0 65535
```

```
Router(config-cmap)# end-class-map
```

```
Router(config)# policy-map type pbr policy1
```

```
Router(config-pmap)# class type traffic class1
```

```
Router(config-pmap-c)# drop
```

```
Router(config-pmap-c)# end
```

Running Configuration

```
hw-module profile flowspec ipv6-packet-len-enable
!
```

```
class-map type traffic match-all class1
  match protocol tcp
  match destination-address ipv6 2:1:1::1/64
  match packet length 0 65535
end-class-map
!
```

```

!
policy-map type pbr policy1
  class type traffic class1
    drop
  end
!
!
!

```

Verification

This example shows sample output from **show flowspec** command when **ipv6** keyword is used to display flowspec policy applied on IPv6 interfaces.

```

Router# show flowspec ipv6 detail

Thu Dec 15 09:51:29.018 UTC

AFI: IPv6
Flow      :Source:193:95::/0-112,TCPFlags:=0x10,Length:>=0&<=65535
Actions   :Traffic-rate: 0 bps (bgp.1)
Statistics (packets/bytes)
  Matched      : 7202356/921901568
  Transmitted   : 0/0
  Dropped      : 7202356/921901568
Flow      :Source:193:96::/0-112,TCPFlags:=0x10,Length:>=0&<=65535
Actions   :Traffic-rate: 0 bps (bgp.1)
Statistics (packets/bytes)
  Matched      : 7203124/950812368
  Transmitted   : 0/0
  Dropped      : 7203124/950812368
Flow      :Source:193:97::/0-112,TCPFlags:=0x10,Length:>=0&<=65535
Actions   :Traffic-rate: 0 bps (bgp.1)
Statistics (packets/bytes)
  Matched      : 7203444/950854608
  Transmitted   : 0/0
  Dropped      : 7203444/950854608
Flow      :Source:193:98::/0-112,TCPFlags:=0x10,Length:>=0&<=65535
Actions   :Traffic-rate: 0 bps (bgp.1)
Statistics (packets/bytes)
  Matched      : 7204032/922116096
  Transmitted   : 0/0
  Dropped      : 7204032/922116096
Flow      :Source:193:99::/0-112,TCPFlags:=0x10,Length:>=0&<=65535
Actions   :Traffic-rate: 0 bps (bgp.1)
Statistics (packets/bytes)
  Matched      : 7202944/950788608
  Transmitted   : 0/0
  Dropped      : 7202944/950788608
-----More-----

```

This example shows sample output from **show flowspec** command when **afi-all** keyword is used to display flowspec policy applied on IPv4 and IPv6 interfaces.

```

Router# show flowspec afi-all detail

Tue Aug 16 08:41:29.893 UTC

AFI: IPv6
Flow
:Dest:193:1::2/0-128,Source:192:1::/0-64,NH:=6,DPort:>=7000&<=20000,SPort:>=7000&<=20000,Length:>=100&<=300,DSCP:=10

```

```

Actions      :DSCP: af21  (policy.1.v6_pm_policymap_set1.v6_cm_1)
Statistics   (packets/bytes)
  Matched    :           0/0
  Transmitted:           0/0
  Dropped    :           0/0
Flow         :DSCP:=18
Actions      :Traffic-rate: 0 bps  (policy.1.v6_pm_policymap_drop1.v6_cm_dscp)
Statistics   (packets/bytes)
  Matched    :           17487/2238336
  Transmitted:           0/0
  Dropped    :           17487/2238336

```

BGP Extended Route Retention

Table 35: 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 attribute. However, this feature ensures that the BGP neighbor considers the stale routes as new routes.

When a BGP peer fails, the Extended Route Retention feature applies the route retention policy to the routes to modify the route attributes. This feature modifies the route attributes in addition to the modification that occur due to neighbor's inbound policy. This feature enables the use of route retention policy in place of LLGR, when the BGP hold timer expires or when the BGP session fails to reestablish as a receiving speaker within the configured graceful restart timer.

When you apply LLGR, you cannot remove the LLGR_STALE community when the stale route is advertised, and the route will treat it as the least preferred. Also, stale routes may be advertised to those neighbors that would not have advertised the LLGR capability under the following conditions:

- The neighbors must be internal (IBGP or confederation) neighbors.
- The NO_EXPORT community must be attached to the stale routes.
- The stale routes must have their LOCAL_PREF community set to zero.

This feature provides you the flexibility to advertise stale routes to eBGP neighbors and enable you to specify local preference values for any stale route that is retained within the iBGP system.

Restrictions

- The neighbor should be capable of graceful restart.
- When the BGP neighbor fails, the graceful restart functionality is applied till the graceful restart timer is valid.
- The Extend Route Retention feature starts, when the graceful restart timer expires,

- Soft-reconfiguration inbound configuration is a mandatory configuration. If required, configure the inbound policy.
- The Extended Route Retention feature starts only when BGP peer goes down, that is, on the expiry of the hold-down timer.
- For any other trigger, such as the expiry of a timer, the routes will not be indicated as stale and the routes is purged.
- The Extended Route Retention feature is applicable only to the following address-family modes:
 - IPv4 and IPv6 unicast address family mode
 - IPv4 and IPv4 labelled unicast address family mode
- You cannot configure both LLGR and Extended Route Retention feature on the same neighbor.
- When you configure the Extended Route Retention feature, the capability attribute is not sent.

Configuration Example

How a CLUSTER_LIST Attribute is Used

The CLUSTER_LIST propagation rules differ among releases, depending on whether the device is running a Cisco software release generated before or after the BGP—Multiple Cluster IDs feature was implemented. The same is true for loop prevention based on the CLUSTER_LIST.

The CLUSTER_LIST behavior is described below. Classic refers to the behavior of software released before the multiple cluster IDs feature was implemented; MCID refers to the behavior of software released after the feature was implemented.

CLUSTER_LIST Propagation Rules

- Classic—Before reflecting a route, the RR appends the global cluster ID to the CLUSTER_LIST. If the received route had no CLUSTER_LIST attribute, the RR creates a new CLUSTER_LIST attribute with that global cluster ID.
- MCID—Before reflecting a route, the RR appends the cluster ID of the neighbor the route was received from to the CLUSTER_LIST. If the received route had no CLUSTER_LIST attribute, the RR creates a new CLUSTER_LIST attribute with that cluster ID. This behavior includes a neighbor that is not a client of the speaker. If the nonclient neighbor the route was received from does not have an associated cluster ID, the RR uses the global cluster ID.

Loop Prevention Based on CLUSTER_LIST

- Classic—When receiving a route, the RR discards the route if the RR's global cluster ID is contained in the CLUSTER_LIST of the route.
- MCID—When receiving a route, the RR discards the route if the RR's global cluster ID or any of the cluster IDs assigned to any of the iBGP neighbors is contained in the CLUSTER_LIST of the route.

Configure a Cluster ID per Neighbor

Perform this task on an iBGP peer, usually a route reflector, to configure a cluster ID per neighbor. Configuring a cluster ID per neighbor causes the loop-prevention mechanism based on the CLUSTER_LIST to be

automatically modified to take into account multiple cluster IDs. Also, you gain the ability to disable client-to-client route reflection on the basis of cluster ID. The software tags the neighbor so that you can disable route reflection with the use of another command.



Note When you change a cluster ID for a neighbor, BGP automatically does an inbound soft refresh and an outbound soft refresh for all iBGP peers.

```
Router> enable
Router # configure terminal
Router(config)# router bgp 65000
Router(config-router)# neighbor 192.168.1.2
Router(config-router)# remote-as 65000
Router(config-router)# cluster-id 0.0.0.1
Router(config-router)# end
```

Running Configuration

```
!
!
router bgp 65000
  neighbor 192.168.1.2
  remote-as 65000
  cluster-id 0.0.0.1
```

Verification

The following example shows that if a cluster-id is configured on any level, either global or per-neighbour, it will be added to the active cluster IDs regardless of the neighbour state. BGP does not track the neighbour state for this feature.

```
Router# show bgp process detail

BGP Process Information:
BGP is operating in STANDALONE mode
Autonomous System number format: ASPLAIN
Autonomous System: 65000
Router ID: 10.10.1.92 (manually configured)
Default Cluster ID: 10.10.1.92
Active Cluster IDs: 10.10.1.92, 10.10.3.93, 10.10.4.20
                   10.10.5.20, 198.51.100.254
...

Router# show configuration commit change last 1

Building configuration...
!! IOS XR Configuration 6.1.3
router bgp 65000
neighbor 198.51.100.254                                <<< not operational, no AFs etc
  remote-as 65000
  cluster-id 198.51.100.254
!
!
end
```


Disable Client-to-Client Reflection for Specified Cluster IDs



Note When the software changes reflection state for a given cluster ID, BGP sends an outbound soft refresh to all clients.

```
Router# configure terminal
Router(config)# router bgp 65000
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# bgp client-to-client reflection cluster-id 0.0.0.1 disable
Router(config-bgp)# commit
```

Running Configuration

```
!
router bgp 65000
  address-family ipv4 unicast
    bgp client-to-client reflection cluster-id 0.0.0.1 disable
```

Verification

The following show command output shows that client-to-client reflection for the cluster IDs has been disabled.

```
Router# show bgp process
BGP Process Information:
BGP is operating in STANDALONE mode
Autonomous System number format: ASPLAIN
Autonomous System: 65000
Router ID: 0.0.0.0
Active Cluster IDs: 0.0.0.1
Fast external fallover enabled
Platform RLIMIT max: 2147483648 bytes
Maximum limit for BMP buffer size: 409 MB
Default value for BMP buffer size: 307 MB
Current limit for BMP buffer size: 307 MB
Current utilization of BMP buffer limit: 0 B
Neighbor logging is enabled
Enforce first AS enabled
Default local preference: 100
Default keepalive: 60
Non-stop routing is enabled
Update delay: 120
Generic scan interval: 60

Address family: IPv4 Unicast
Dampening is not enabled
Client reflection is not 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: 0
Prefixes scanned per segment: 100000
Number of scan segments: 1
Nexthop resolution minimum prefix-length: 0 (not configured)
Main Table Version: 2
Table version synced to RIB: 2
Table version acked by RIB: 2
IGP notification: IGP notified
```

```

RIB has converged: version 0
RIB table prefix-limit reached ? [No], version 0
Permanent Network Unconfigured

Node          Process      Nbrs Estb Rst Upd-Rcvd Upd-Sent Nfn-Rcv Nfn-Snt
node0_0_CPU0  Speaker      1     0   2      0       0       0       3

```

How to Implement BGP

Information About Implementing BGP

To implement BGP, you need to understand the following concepts:

Adjust BGP Timers

BGP uses certain timers to control periodic activities, such as the sending of keepalive messages and the interval after which a neighbor is assumed to be down if no messages are received from the neighbor during the interval. The values set using the **timers bgp** command in router configuration mode can be overridden on particular neighbors using the **timers** command in the neighbor configuration mode.

Perform this task to set the timers for BGP neighbors.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 123
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **timers bgp** *keepalive hold-time*

Example:

```
Router(config-bgp)# timers bgp 30 90
```

Sets a default keepalive time and a default hold time for all neighbors.

Step 4 **neighbor** *ip-address*

Example:

```
Router(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 5 `timers keepalive hold-time`

Example:

```
Router(config-bgp-nbr)# timers 60 220
```

(Optional) Sets the keepalive timer and the hold-time timer for the BGP neighbor.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Enabling BGP Routing

Perform this task to enable BGP routing and establish a BGP routing process. Configuring BGP neighbors is included as part of enabling BGP routing.



Note At least one neighbor and at least one address family must be configured to enable BGP routing. At least one neighbor with both a remote AS and an address family must be configured globally using the **address family** and **remote as** commands.

Before you begin

BGP must be able to obtain a router identifier (for example, a configured loopback address). At least, one address family must be configured in the BGP router configuration and the same address family must also be configured under the neighbor.



Note If the neighbor is configured as an external BGP (eBGP) peer, you must configure an inbound and outbound route policy on the neighbor using the **route-policy** command.



Note Instead of configuring an inbound and outbound route policy, you can configure the unsafe eBGP policy to allow all eBGP neighbors to pass routes using the **bgp unsafe-ebgp-policy** command.



Note While establishing eBGP neighborship between two peers, BGP checks if the two peers are directly connected. If the peers are not directly connected, BGP does not try to establish a relationship by default. If two BGP peers are not directly connected and peering is required between the loop backs of the routers, you can use the **ignore-connected-check** command. This command overrides the default check that BGP performs which is to verify if source IP in BGP control packets is in same network as that of destination. In this scenario, a TTL value of 1 is sufficient if **ignore-connected-check** is used.

Configuring **egp-multihop ttl** is needed when the peers are not directly connected and there are more routers in between. If the **egp-multihop ttl** command is not configured, eBGP sets the TTL of packets carrying BGP messages to 1 by default. When eBGP needs to be setup between routers which are more than one hop away, you need to configure a TTL value which is at least equal to the number of hops between them. For example, if there are 2 hops (R2, R3) between two BGP peering routers R1 and R4, you need to set a TTL value of 3.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **route-policy route-policy-name**

Example:

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

(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.

Step 3 **end-policy**

Example:

```
Routing(config-rpl)# end-policy
```

(Optional) Ends the definition of a route policy and exits route policy configuration mode.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.

- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 5 **configure****Example:**

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 6 **router bgp** *as-number***Example:**

```
Routing(config)# router bgp 120
```

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 7 **bgp router-id** *ip-address***Example:**

```
Routing(config-bgp)# bgp router-id 192.168.70.24
```

Configures the local router with a specified router ID.

Step 8 **address-family** { **ipv4** | **ipv6** } **unicast****Example:**

```
Routing(config-bgp)# address-family ipv4 unicast
```

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 9 **exit****Example:**

```
Routing(config-bgp-af)# exit
```

Exits the current configuration mode.

Step 10 **neighbor** *ip-address***Example:**

```
Routing(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 11 **remote-as** *as-number***Example:**

```
Routing(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 12 **address-family** { **ipv4** | **ipv6** } **unicast**

Example:

```
Routing(config-bgp-nbr)# address-family ipv4 unicast
```

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 13 **route-policy** *route-policy-name* { **in** | **out** }

Example:

```
Routing(config-bgp-nbr-af)# route-policy drop-as-1234 in
```

(Optional) Applies the specified policy to inbound IPv4 unicast routes.

Step 14 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure Multiple BGP Instances for a Specific Autonomous System

Perform this task to configure multiple BGP instances for a specific autonomous system. All configuration changes for a single BGP instance can be committed together. However, configuration changes for multiple instances cannot be committed together.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number* [**instance** *instance name*]
3. **bgp router-id** *ip-address*
4. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number* [**instance** *instance name*]

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 100 instance inst1
```

Enters BGP configuration mode for the user specified BGP instance.

Step 3 **bgp router-id** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 10.0.0.0
```

Configures a fixed router ID for the BGP-speaking router (BGP instance).

Note

You must manually configure unique router ID for each BGP instance.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure Routing Domain Confederation for BGP

Perform this task to configure the routing domain confederation for BGP. This includes specifying a confederation identifier and autonomous systems that belong to the confederation.

Configuring a routing domain confederation reduces the internal BGP (iBGP) mesh by dividing an autonomous system into multiple autonomous systems and grouping them into a single confederation. Each autonomous system is fully meshed within itself and has a few connections to another autonomous system in the same confederation. The confederation maintains the next hop and local preference information, and that allows you to retain a single Interior Gateway Protocol (IGP) for all autonomous systems. To the outside world, the confederation looks like a single autonomous system.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:

```
Router# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **bgp confederation identifier** *as-number*

Example:

```
Router(config-bgp)# bgp confederation identifier 5
```

Specifies a BGP confederation identifier.

Step 4 **bgp confederation peers** *as-number*

Example:

```
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
Router(config-bgp)# bgp confederation peers 1095
Router(config-bgp)# bgp confederation peers 1096
```

Specifies that the BGP autonomous systems belong to a specified BGP confederation identifier. You can associate multiple AS numbers to the same confederation identifier, as shown in the example.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

BGP Confederation: Example

The following is a sample configuration that shows several peers in a confederation. The confederation consists of three internal autonomous systems with autonomous system numbers 6001, 6002, and 6003. To the BGP speakers outside the confederation, the confederation looks like a normal autonomous system with autonomous system number 666 (specified using the **bgp confederation identifier** command).

In a BGP speaker in autonomous system 6001, the **bgp confederation peers** command marks the peers from autonomous systems 6002 and 6003 as special eBGP peers. Hence, peers 171.16 .232.55 and 171.16 .232.56 get the local preference, next hop, and MED unmodified in the updates. The

router at 171.19.69.1 is a normal eBGP speaker, and the updates received by it from this peer are just like a normal eBGP update from a peer in autonomous system 666.

```
router bgp 6001
  bgp confederation identifier 666
  bgp confederation peers
    6002
    6003
  exit
  address-family ipv4 unicast
    neighbor 171.16.232.55
    remote-as 6002
  exit
  address-family ipv4 unicast
    neighbor 171.16.232.56
    remote-as 6003
  exit
  address-family ipv4 unicast
    neighbor 171.19.69.1
    remote-as 777
```

In a BGP speaker in autonomous system 6002, the peers from autonomous systems 6001 and 6003 are configured as special eBGP peers. Peer 171.17.70.1 is a normal iBGP peer, and peer 199.99.99.2 is a normal eBGP peer from autonomous system 700.

```
router bgp 6002
  bgp confederation identifier 666
  bgp confederation peers
    6001
    6003
  exit
  address-family ipv4 unicast
    neighbor 171.17.70.1
    remote-as 6002
  exit
  address-family ipv4 unicast
    neighbor 171.19.232.57
    remote-as 6001
  exit
  address-family ipv4 unicast
    neighbor 171.19.232.56
    remote-as 6003
  exit
  address-family ipv4 unicast
    neighbor 171.19.99.2
    remote-as 700
  exit
  address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-all out
```

In a BGP speaker in autonomous system 6003, the peers from autonomous systems 6001 and 6002 are configured as special eBGP peers. Peer 192.168.200.200 is a normal eBGP peer from autonomous system 701.

```
router bgp 6003
  bgp confederation identifier 666
```

```

bgp confederation peers
 6001
 6002
 exit
address-family ipv4 unicast
 neighbor 171.19.232.57
  remote-as 6001
 exit
address-family ipv4 unicast
 neighbor 171.19.232.55
  remote-as 6002
 exit
address-family ipv4 unicast
 neighbor 192.168.200.200
  remote-as 701
 exit
address-family ipv4 unicast
 route-policy pass-all in
 route-policy pass-all out

```

The following is a part of the configuration from the BGP speaker 192.168.200.205 from autonomous system 701 in the same example. Neighbor 171.16.232.56 is configured as a normal eBGP speaker from autonomous system 666. The internal division of the autonomous system into multiple autonomous systems is not known to the peers external to the confederation.

```

router bgp 701
 address-family ipv4 unicast
  neighbor 172.16.232.56
  remote-as 666
 exit
address-family ipv4 unicast
 route-policy pass-all in
 route-policy pass-all out
 exit
address-family ipv4 unicast
 neighbor 192.168.200.205
  remote-as 701

```

Resetting an eBGP Session Immediately Upon Link Failure

By default, if a link goes down, all BGP sessions of any directly adjacent external peers are immediately reset. Use the **bgp fast-external-fallover disable** command to disable automatic resetting. Turn the automatic reset back on using the **no bgp fast-external-fallover disable** command.

eBGP sessions flap when the node reaches 3500 eBGP sessions with BGP timer values set as 10 and 30. To support more than 3500 eBGP sessions, increase the packet rate by using the **lpts pifib hardware police location location-id** command. Following is a sample configuration to increase the eBGP sessions:

```

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

```

BGP Fast Fallover

Table 36: Feature History Table

Feature Name	Release Information	Feature Description
BGP Fast Fallover	Release 24.2.11	<p>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-fallover disable command is globally configured.</p> <p>This feature enables quicker failure detection, and allows other recovery mechanisms to reroute the traffic quickly, thus resulting in faster convergence.</p> <p>The feature introduces these changes:</p> <p>CLI:</p> <ul style="list-style-type: none">• fast-fallover <p>YANG Data Model:</p> <ul style="list-style-type: none">• Cisco-IOS-XR-um-router-bgp-cfg.yang (see GitHub, YANG Data Models Navigator)

Faster Network Convergence with Fast Fallover

When an interface attached to a directly connected BGP neighbor fails, the routes learned from that neighbor persist until the hold timer expires. This lag can lead to slow network convergence and potential network instability.

By introducing a fast fallover mechanism, this feature ensures that when an IP interface fails, routes learned from a directly connected iBGP or eBGP neighbor are quickly removed from the BGP routing table, without having to wait for the hold timer to expire. This accelerates the network convergence process and prevents the propagation of potentially stale routes.



Note You can also use the **nexthop trigger-delay** command to quickly remove the BGP routes of a failing neighbor, provided that the neighbor's BGP session endpoint is the same as the route's next hop.

Limitations of BGP Fast Fallover

These limitations apply to the BGP fast fallover feature:

- Fast fallover is applicable only to directly connected BGP neighbors. A directly connected BGP neighbor can either be a neighbor one hop away, or the neighbor's IP address is within the same subnet. BGP neighbor connected through a loopback interface is not considered as directly connected, even if it's just one hop away. The neighbors that are not directly connected retain their current behavior, that is,

maintaining the established connection until a triggering event, such as the expiration of a hold timer, brings the session down.

- If an interface fails before fast fallover is activated, then the session to the connected neighbor will not go down. So, you must manually clear the BGP neighbor session, if necessary.
- Regular BGP session establishment processes remain unchanged when an interface recovers from failure.

Configure BGP Fast-Fallover

Procedure

Step 1 Enable fast fallover mechanism using the **fast-fallover** command under BGP neighbor, neighbor group, or session group configuration modes.

- By default, fast fallover is enabled for eBGP neighbors and disabled for iBGP neighbors. If the **bgp fast-external-fallover disable** command is configured under BGP global or VRF configuration modes, fast fallover will be disabled for eBGP neighbors, but it can still be overridden for the specific eBGP neighbors using the **fast-fallover** command.

Example:

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# neighbor 209.165.201.0
Router(config-bgp-nbr)# fast-fallover
```

Step 2 To prevent the fast fallover setting from being inherited from a higher-level neighbor group or session group, use the **fast-fallover inheritance-disable** command.

Step 3 Use the **show bgp neighbors ip-address** or **show run router bgp as-number neighbor ip-address** command to verify if fast fallover is enabled for the neighbor or inherited from a neighbor group or session group.

Example:

```
Router# show bgp neighbors 209.165.201.0
```

```
BGP neighbor is 209.165.201.0
 Remote AS 1, local AS 1, internal link
 Remote router ID 10.0.0.0
 Cluster ID 10.1.1.1
 BGP state = Active
 Previous State: Idle
 Last Received Message: KeepAlive
 NSR State: None
 Last read 00:00:00, Last read before reset 00:46:59
 Hold time is 90, keepalive interval is 30 seconds
 Configured hold time: 90, keepalive: 30, min acceptable hold time: 3
 Last write 00:46:56, attempted 19, written 19
 Second last write 00:47:26, attempted 19, written 19
 Last write before reset 00:46:56, attempted 19, written 19
 Second last write before reset 00:47:26, attempted 19, written 19
 Last write pulse rcvd May 6 17:05:39.083 last full not set pulse count 207
 Last write pulse rcvd before reset 00:46:56
 Last insert into reset queue: May 6 17:05:45.532, removed at May 6 17:05:45.532
 Socket not armed for io, not armed for read, not armed for write
 Last write thread event before reset 00:46:56, second last 00:46:56
 Last KA expiry before reset 00:46:56, second last 00:47:26
```

```

Last KA error before reset 00:00:00, KA not sent 00:00:00
Last KA start before reset 00:46:56, second last 00:47:26
Precedence: internet
Non-stop routing is enabled
BGP Monitoring(BMP) activated for servers:
    1, 2, 3, 4
Multi-protocol capability received
Received 104 messages, 0 notifications, 0 in queue
Sent 103 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
Fast fallover is enabled
    Neighbor is directly connected
    Neighbor fast-fallover is configured
    Neighbor is external and fast-external-fallover is not disabled

For Address Family: IPv4 Unicast
BGP neighbor version 0
Update group: 0.21 Filter-group: 0.0 No Refresh request being processed
Route-Reflector Client

```

The details **Fast fallover is (not) enabled**, **Neighbor is (not) directly connected**, and **Neighbor fast-fallover is (not) configured**, indicate the state of the fast fallover feature. The **Neighbor is external and fast-external-fallover is not disabled** line is displayed only for external neighbors.

Example:

```
Router# show run router bgp 1 neighbor 209.165.201.0
```

```

router bgp 1
neighbor 209.165.201.0
  remote-as 1
  bmp-activate server 1
  bmp-activate server 2
  bmp-activate server 3
  bmp-activate server 4
  fast-fallover
  address-family ipv4 unicast
    use af-group ipv4_unicast_3_ibgp
  !
!
!

```

fast-fallover displayed in the output shows that the fast fallover feature is successfully configured for the neighbor.

Logging Neighbor Changes

Logging neighbor changes is enabled by default. Use the **log neighbor changes disable** command to turn off logging. The **no log neighbor changes disable** command can also be used to turn logging back on if it has been disabled.

Change BGP Default Local Preference Value

Perform this task to set the default local preference value for BGP paths.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **bgp default local-preference** *value*

Example:

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

Sets the default local preference value from the default of 100, making it either a more preferable path (over 100) or less preferable path (under 100).

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure MED Metric for BGP

Perform this task to set the multi exit discriminator (MED) to advertise to peers for routes that do not already have a metric set (routes that were received with no MED attribute).

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:

```
Routing(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **default-metric** *value*

Example:

```
Routing(config-bgp)# default metric 10
```

Sets the default metric, which is used to set the MED to advertise to peers for routes that do not already have a metric set (routes that were received with no MED attribute).

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure BGP Weights

A weight is a number that you can assign to a path so that you can control the best-path selection process. If you have particular neighbors that you want to prefer for most of your traffic, you can use the **weight** command to assign a higher weight to all routes learned from that neighbor. Perform this task to assign a weight to routes received from a neighbor.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:

```
Routing(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address***Example:**

```
Routing(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **remote-as** *as-number***Example:**

```
Routing(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 5 **address-family** { **ipv4** | **ipv6** } **unicast****Example:**

```
Routing(config-bgp-nbr)# address-family ipv4 unicast
```

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 6 **weight** *weight-value***Example:**

```
Routing(config-bgp-nbr-af)# weight 41150
```

Assigns a weight to all routes learned through the neighbor.

Step 7 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

You the **clear bgp** command for the newly configured weight to take effect.

Tune BGP Best-Path Calculation

BGP routers typically receive multiple paths to the same destination. The BGP best-path algorithm determines the best path to install in the IP routing table and to use for forwarding traffic. The BGP best-path comprises of three steps:

- Step 1—Compare two paths to determine which is better.
- Step 2—Iterate over all paths and determines which order to compare the paths to select the overall best path.
- Step 3—Determine whether the old and new best paths differ enough so that the new best path should be used.



Note The order of comparison determined by Step 2 is important because the comparison operation is not transitive; that is, if three paths, A, B, and C exist, such that when A and B are compared, A is better, and when B and C are compared, B is better, it is not necessarily the case that when A and C are compared, A is better. This nontransitivity arises because the multi exit discriminator (MED) is compared only among paths from the same neighboring autonomous system (AS) and not among all paths.

Perform this task to change the default BGP best-path calculation behavior.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp *as-number***

Example:

```
Router(config)# router bgp 126
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **bgp bestpath med missing-as-worst**

Example:

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

Directs the BGP software to consider a missing MED attribute in a path as having a value of infinity, making this path the least desirable path.

Step 4 **bgp bestpath med always**

Example:

```
Router(config-bgp)# bgp bestpath med always
```

Configures the BGP speaker in the specified autonomous system to compare MEDs among all the paths for the prefix, regardless of the autonomous system from which the paths are received.

Step 5 **bgp bestpath med confed**

Example:

```
Router(config-bgp)# bgp bestpath med confed
```

Enables BGP software to compare MED values for paths learned from confederation peers.

Step 6 **bgp bestpath as-path ignore**

Example:

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

Configures the BGP software to ignore the autonomous system length when performing best-path selection.

Step 7 **bgp bestpath compare-routerid**

Example:

```
Router(config-bgp)# bgp bestpath compare-routerid
```

Configure the BGP speaker in the autonomous system to compare the router IDs of similar paths.

Step 8 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure Aggregate Addresses

Perform this task to create aggregate entries in a BGP routing table.



Note

For optimal CPU utilization when deploying BGP aggregate for supernet addresses with a higher scale such as internet bgp table, it is recommended to:

- Use aggregate subnet of size not exceeding /24.
- Tune the subnet mask size based on network scale and churn.
- Use the **default-originate** or **network** 0.0.0.0 CLI instead of 0.0.0.0 as aggregate, when advertising the default route 0.0.0.0.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp *as-number***

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family { *ipv4* | *ipv6* } unicast**

Example:

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

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 4 **aggregate-address *address/mask-length* [*as-set*] [*as-confed-set*] [*summary-only*] [*route-policy route-policy-name*]**

Example:

```
Router(config-bgp-af)# aggregate-address 10.0.0.0/8 as-set
```

Creates an aggregate address. The path advertised for this route is an autonomous system set consisting of all elements contained in all paths that are being summarized.

- The **as-set** keyword generates autonomous system set path information and community information from contributing paths.
- The **as-confed-set** keyword generates autonomous system confederation set path information from contributing paths.
- The **summary-only** keyword filters all more specific routes from updates.
- The **route-policy *route-policy-name*** keyword and argument specify the route policy used to set the attributes of the aggregate route.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.

- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Indicate BGP Back-door Routes

Perform this task to set the administrative distance on an external Border Gateway Protocol (eBGP) route to that of a locally sourced BGP route, causing it to be less preferred than an Interior Gateway Protocol (IGP) route.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp *as-number***

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family { *ipv4* | *ipv6* } unicast**

Example:

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

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 4 **network { *ip-address / prefix-length* | *ip-address mask* } backdoor**

Example:

```
Router(config-bgp-af)# network 172.20.0.0/16
```

Configures the local router to originate and advertise the specified network.

Step 5 Use the **commit** or **end** command.

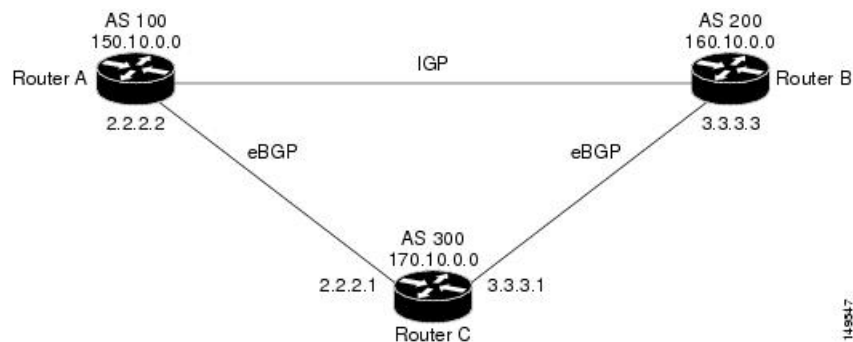
commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.

- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Back Door: Example



Here, Routers A and C and Routers B and C are running eBGP. Routers A and B are running an IGP (such as Routing Information Protocol [RIP], Interior Gateway Routing Protocol [IGRP], Enhanced IGRP, or Open Shortest Path First [OSPF]). The default distances for RIP, IGRP, Enhanced IGRP, and OSPF are 120, 100, 90, and 110, respectively. All these distances are higher than the default distance of eBGP, which is 20. Usually, the route with the lowest distance is preferred.

Router A receives updates about 160.10.0.0 from two routing protocols: eBGP and IGP. Because the default distance for eBGP is lower than the default distance of the IGP, Router A chooses the eBGP-learned route from Router C. If you want Router A to learn about 160.10.0.0 from Router B (IGP), establish a BGP back door. See .

In the following example, a network back-door is configured:

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

Router A treats the eBGP-learned route as local and installs it in the IP routing table with a distance of 200. The network is also learned through Enhanced IGRP (with a distance of 90), so the Enhanced IGRP route is successfully installed in the IP routing table and is used to forward traffic. If the Enhanced IGRP-learned route goes down, the eBGP-learned route is installed in the IP routing table and is used to forward traffic.

Although BGP treats network 160.10.0.0 as a local entry, it does not advertise network 160.10.0.0 as it normally would advertise a local entry.

Set BGP Administrative Distance

An administrative distance is a rating of the trustworthiness of a routing information source. In general, the higher the value, the lower the trust rating. Normally, a route can be learned through more than one protocol. Administrative distance is used to discriminate between routes learned from more than one protocol. The route with the lowest administrative distance is installed in the IP routing table. By default, BGP uses the administrative distances shown in here:

Table 37: 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.



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

Perform this task to specify the use of administrative distances that can be used to prefer one class of route over another.

Procedure

Step 1 **configure**

Step 2 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family { ipv4 | ipv6 } unicast**

Example:

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submenu.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 4 **distance bgp** *external-distance internal-distance local-distance*

Example:

```
Router(config-bgp-af)# distance bgp 20 20 200
```

Sets the external, internal, and local administrative distances to prefer one class of routes over another. The higher the value, the lower the trust rating.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** — Exits the configuration session without committing the configuration changes.
- **Cancel** — Remains in the configuration session, without committing the configuration changes.

Configure BGP Neighbor Group and Neighbors

Perform this task to configure BGP neighbor groups and apply the neighbor group configuration to a neighbor. A neighbor group is a template that holds address family-independent and address family-dependent configurations associated with the neighbor.

After a neighbor group is configured, each neighbor can inherit the configuration through the **use** command. If a neighbor is configured to use a neighbor group, the neighbor (by default) inherits the entire configuration of the neighbor group, which includes the address family-independent and address family-dependent configurations. The inherited configuration can be overridden if you directly configure commands for the neighbor or configure session groups or address family groups through the **use** command.

You can configure an address family-independent configuration under the neighbor group. An address family-dependent configuration requires you to configure the address family under the neighbor group to enter address family submode. From neighbor group configuration mode, you can configure address family-independent parameters for the neighbor group. Use the **address-family** command when in the neighbor group configuration mode. After specifying the neighbor group name using the **neighbor group** command, you can assign options to the neighbor group.



Note All commands that can be configured under a specified neighbor group can be configured under a neighbor.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family** { **ipv4** | **ipv6** } **unicast**

Example:

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 4

exit

Example:

```
Router(config-bgp-af) # exit
```

Exits the current configuration mode.

Step 5

neighbor-group *name*

Example:

```
Router(config-bgp) # neighbor-group nbr-grp-A
```

Places the router in neighbor group configuration mode.

Step 6

remote-as *as-number*

Example:

```
Router(config-bgp-nbrgrp) # remote-as 2002
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 7

address-family { **ipv4** | **ipv6** } **unicast**

Example:

```
Router(config-bgp-nbrgrp) # address-family ipv4 unicast
```

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 8

route-policy *route-policy-name* { **in** | **out** }

Example:

```
Router(config-bgp-nbrgrp-af) # route-policy drop-as-1234 in
```

(Optional) Applies the specified policy to inbound IPv4 unicast routes.

Step 9

exit

Example:

```
Router(config-bgp-nbrgrp-af) # exit
```

Exits the current configuration mode.

Step 10

exit

Example:

```
Router(config-bgp-nbrgrp) # exit
```

Exits the current configuration mode.

Step 11

neighbor *ip-address*

Example:

```
Router(config-bgp) # neighbor 172.168.40.24
```


Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 12 **use neighbor-group** *group-name*

Example:

```
Router(config-bgp-nbr)# use neighbor-group nbr-grp-A
```

(Optional) Specifies that the BGP neighbor inherit configuration from the specified neighbor group.

Step 13 **remote-as** *as-number*

Example:

```
Router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 14 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

BGP Neighbor Configuration: Example

The following example shows how BGP neighbors on an autonomous system are configured to share information. In the example, a BGP router is assigned to autonomous system 109, and two networks are listed as originating in the autonomous system. Then the addresses of three remote routers (and their autonomous systems) are listed. The router being configured shares information about networks 172.16.0.0 and 192.168.1.7.0 with the neighbor routers. The first router listed is in a different autonomous system; the second **neighbor** and **remote-as** commands specify an internal neighbor (with the same autonomous system number) at address 172.26.234.2; and the third **neighbor** and **remote-as** commands specify a neighbor on a different autonomous system.

```
route-policy pass-all
pass
end-policy
router bgp 109
address-family ipv4 unicast
network 172.16.0.0 255.255.0.0
network 192.168.1.7.0 255.255.0.0
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
```

Configure Route Reflector for BGP

Perform this task to configure a route reflector for BGP.

All the neighbors configured with the **route-reflector-client** command are members of the client group, and the remaining iBGP peers are members of the nonclient group for the local route reflector.

Together, a route reflector and its clients form a *cluster*. A cluster of clients usually has a single route reflector. In such instances, the cluster is identified by the software as the router ID of the route reflector. To increase redundancy and avoid a single point of failure in the network, a cluster can have more than one route reflector. If it does, all route reflectors in the cluster must be configured with the same 4-byte cluster ID so that a route reflector can recognize updates from route reflectors in the same cluster. The **bgp cluster-id** command is used to configure the cluster ID when the cluster has more than one route reflector.

The **bgp cluster-id** option is used in this task to configure the router as one of the route reflectors serving the cluster. The **cluster-id** option is also available in the BGP neighbor address-family (config-bgp-nbr-af) mode. To enable a router to accept BGP routes which have the same first cluster-ID as the router's own cluster-ID in the list of cluster-IDs, use the **cluster-id allow-equal** command. You must use this command with care to avoid routing loops.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
Enters mode.
```

Step 2 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **bgp cluster-id** *cluster-id*

Example:

```
Router(config-bgp)# bgp cluster-id 192.168.70.1
```

Configures the local router as one of the route reflectors serving the cluster. It is configured with a specified cluster ID to identify the cluster.

Step 4 **neighbor** *ip-address***Example:**

```
Router(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 5 **remote-as** *as-number***Example:**

```
Router(config-bgp-nbr)# remote-as 2003
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 6 **address-family** { **ipv4** | **ipv6** } **unicast****Example:**

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 7 **route-reflector-client****Example:**

```
Router(config-bgp-nbr-af)# route-reflector-client
```

Configures the router as a BGP route reflector and configures the neighbor as its client.

Step 8 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

BGP Route Reflector: Example

The following example shows how to use an address family to configure internal BGP peer 10.1.1.1 as a route reflector client for unicast prefixes:

```
router bgp 140
 address-family ipv4 unicast
  neighbor 10.1.1.1
  remote-as 140
 address-family ipv4 unicast
  route-reflector-client
 exit
```

Understanding BGP MD5 Authentication

Table 38: Feature History Table

Feature Name	Release Name	Description
Understanding BGP MD5 Authentication	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700) (select variants only*)</p> <p>This feature enhances security by authenticating TCP segments between BGP peers using MD5 passwords. Both peers must configure the same password at the BGP neighbor level. If authentication fails, packets are blocked, preventing unauthorized access.</p> <p>*BGP MD5 Authentication is now supported on Cisco 8712-MOD-M routers.</p>

BGP provides a mechanism, known as Message Digest 5 (MD5) authentication, for authenticating a TCP segment between two BGP peers by using a clear text or encrypted password.

MD5 authentication is configured at the BGP neighbor level. BGP peers using MD5 authentication are configured with the same password. If the password authentication fails, then the packets are not transmitted along the segment.

Redistributing iBGP Routes into IGP

Perform this task to redistribute iBGP routes into an Interior Gateway Protocol (IGP), such as Intermediate System-to-Intermediate System (IS-IS) or Open Shortest Path First (OSPF).



Note Use of the **bgp redistribute-internal** command requires the **clear route *** command to be issued to reinstall all BGP routes into the IP routing table.



Caution Redistributing iBGP routes into IGP may cause routing loops to form within an autonomous system. Use this command with caution.

Procedure

	Command or Action	Purpose
Step 1	configure	

	Command or Action	Purpose
Step 2	router bgp <i>as-number</i> Example: <pre>Router(config)# router bgp 120</pre>	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	bgp redistribute-internal Example: <pre>Router(config-bgp)# bgp redistribute-internal</pre>	Allows the redistribution of iBGP routes into an IGP, such as IS-IS or OSPF.
Step 4	commit	

Set BGP Administrative Distance

An administrative distance is a rating of the trustworthiness of a routing information source. In general, the higher the value, the lower the trust rating. Normally, a route can be learned through more than one protocol. Administrative distance is used to discriminate between routes learned from more than one protocol. The route with the lowest administrative distance is installed in the IP routing table. By default, BGP uses the administrative distances shown in here:

Table 39: 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.



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

Perform this task to specify the use of administrative distances that can be used to prefer one class of route over another.

Procedure

Step 1 **configure**

Step 2 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family { ipv4 | ipv6 } unicast**

Example:

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 4 **distance bgp external-distance internal-distance local-distance**

Example:

```
Router(config-bgp-af)# distance bgp 20 20 200
```

Sets the external, internal, and local administrative distances to prefer one class of routes over another. The higher the value, the lower the trust rating.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Discard Extra Paths

Perform this task to configure BGP maximum-prefix discard extra paths.

Procedure

	Command or Action	Purpose
Step 1	configure Example: Router# configure	Enters Global configuration mode.
Step 2	router bgp as-number Example: Router(config)# router bgp 10	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

	Command or Action	Purpose
Step 3	neighbor <i>ip-address</i> Example: Router(config-bgp)# neighbor 10.0.0.1	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	address-family { ipv4 ipv6 } unicast Example: Router(config-bgp-nbr)# address-family ipv4 unicast	Specifies either the IPv4 or IPv6 address family and enters address family configuration submenu.
Step 5	maximum-prefix <i>maximum</i> discard-extra-paths Example: Router(config-bgp-nbr-af)# maximum-prefix 1000 discard-extra-paths	Configures a limit to the number of prefixes allowed. Configures discard extra paths to discard extra paths when the maximum prefix limit is exceeded.
Step 6	Use the commit or end command.	commit —Saves the configuration changes and remains within the configuration session. end —Prompts user to take one of these actions: <ul style="list-style-type: none"> • Yes — Saves configuration changes and exits the configuration session. • No —Exits the configuration session without committing the configuration changes. • Cancel —Remains in the configuration session, without committing the configuration changes.

Configuring Per Neighbor TCP MSS

Perform this task to configure TCP MSS under neighbor group, which is inherited by a neighbor.

Procedure

Step 1	configure Example: Router# configure Enters XR Config mode.
Step 2	router bgp <i>as-number</i> Example: Router(config)# router bgp 10 Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family ipv4 unicast****Example:**

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

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 4 **exit****Example:**

```
Router(config-bgp-af)# exit
```

Exits router address family configuration mode, and returns to BGP configuration mode.

Step 5 **neighbor-group name****Example:**

```
Router(config-bgp)# neighbor-group n1
```

Enters neighbor group configuration mode.

Step 6 **tcp mss segment-size****Example:**

```
Router(config-bgp-nbrgrp)# tcp mss 500
```

Configures TCP maximum segment size. The range is from 68 to 10000.

Step 7 **address-family ipv4 unicast****Example:**

```
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
```

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 8 **exit****Example:**

```
Router(config-bgp-nbrgrp-af)# exit
```

Exits router address family configuration mode.

Step 9 **exit****Example:**

```
Router(config-bgp-nbrgrp)# exit
```

Exits the neighbor group configuration mode.

Step 10 **neighbor ip-address****Example:**

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

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 11 **remote-as as-number**

Example:

```
Router(config-bgp-nbr)# remote-as 1
```

Creates a neighbor and assigns a remote autonomous system (AS) number to it.

- Range for 2-byte autonomous system numbers (ASNs) is 1 to 65535.
- Range for 4-byte autonomous system numbers (ASNs) in asplain format is 1 to 4294967295.
- Range for 4-byte autonomous system numbers (ASNs) in asdot format is 1.0 to 65535.65535.

Step 12 **use neighbor-group** *group-name***Example:**

```
Router(config-bgp-nbr)# use neighbor-group n1
```

Specifies that the BGP neighbor inherit configuration from the specified neighbor group.

Step 13 **address-family** **ipv4 unicast****Example:**

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

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

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 14 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Disabling Per Neighbor TCP MSS

Perform this task to disable TCP MSS for a particular neighbor under neighbor group.

Procedure

Step 1 **configure****Example:**

```
Router# configure
```

Step 2 **router bgp** *as-number***Example:**

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

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family ipv4 unicast**

Example:

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

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 4 **exit**

Example:

```
Router(config-bgp-af)# exit
```

Exits router address family configuration mode, and returns to BGP configuration mode.

Step 5 **neighbor-group name**

Example:

```
Router(config-bgp)# neighbor-group n1
```

Enters neighbor group configuration mode.

Step 6 **tcp mss segment-size**

Example:

```
Router(config-bgp-nbrgrp)# tcp mss 500
```

Configures TCP maximum segment size. The range is from 68 to 10000.

Step 7 **address-family ipv4 unicast**

Example:

```
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
```

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 8 **exit**

Example:

```
Router(config-bgp-nbrgrp-af)# exit
```

Exits router address family configuration mode.

Step 9 **exit**

Example:

```
Router(config-bgp-nbrgrp)# exit
```

Exits the neighbor group configuration mode.

Step 10 **neighbor ip-address**

Example:

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

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 11 **remote-as** *as-number*

Example:

```
Router(config-bgp-nbr)# remote-as 1
```

Creates a neighbor and assigns a remote autonomous system (AS) number to it.

- Range for 2-byte autonomous system numbers (ASNs) is 1 to 65535.
- Range for 4-byte autonomous system numbers (ASNs) in asplain format is 1 to 4294967295.
- Range for 4-byte autonomous system numbers (ASNs) in asdot format is 1.0 to 65535.65535.

Step 12 **use neighbor-group** *group-name*

Example:

```
Router(config-bgp-nbr)# use neighbor-group n1
```

Specifies that the BGP neighbor inherit configuration from the specified neighbor group.

Step 13 **tcp mss inheritance-disable**

Example:

```
Router(config-bgp-nbr)# tcp mss inheritance-disable
```

Disables TCP MSS for the neighbor.

Step 14 **address-family ipv4 unicast**

Example:

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

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

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 15 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Discard Extra Paths

Perform this task to configure BGP maximum-prefix discard extra paths.

Procedure

	Command or Action	Purpose
Step 1	configure Example: Router# configure	Enters Global configuration mode.
Step 2	router bgp <i>as-number</i> Example: Router(config)# router bgp 10	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	neighbor <i>ip-address</i> Example: Router(config-bgp)# neighbor 10.0.0.1	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	address-family { <i>ipv4</i> <i>ipv6</i> } unicast Example: Router(config-bgp-nbr)# address-family ipv4 unicast	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.
Step 5	maximum-prefix <i>maximum</i> discard-extra-paths Example: Router(config-bgp-nbr-af)# maximum-prefix 1000 discard-extra-paths	Configures a limit to the number of prefixes allowed. Configures discard extra paths to discard extra paths when the maximum prefix limit is exceeded.
Step 6	Use the commit or end command.	commit —Saves the configuration changes and remains within the configuration session. end —Prompts user to take one of these actions: <ul style="list-style-type: none"> • Yes — Saves configuration changes and exits the configuration session. • No —Exits the configuration session without committing the configuration changes. • Cancel —Remains in the configuration session, without committing the configuration changes.

Configuring Per Neighbor TCP MSS

Perform this task to configure TCP MSS under neighbor group, which is inherited by a neighbor.

Procedure

Step 1 **configure****Example:**

```
Router# configure
```

Enters XR Config mode.

Step 2 **router bgp *as-number*****Example:**

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

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family *ipv4 unicast*****Example:**

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

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 4 **exit****Example:**

```
Router(config-bgp-af)# exit
```

Exits router address family configuration mode, and returns to BGP configuration mode.

Step 5 **neighbor-group *name*****Example:**

```
Router(config-bgp)# neighbor-group n1
```

Enters neighbor group configuration mode.

Step 6 **tcp mss *segment-size*****Example:**

```
Router(config-bgp-nbrgrp)# tcp mss 500
```

Configures TCP maximum segment size. The range is from 68 to 10000.

Step 7 **address-family *ipv4 unicast*****Example:**

```
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
```

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 8 **exit****Example:**

```
Router(config-bgp-nbrgrp-af) # exit
```

Exits router address family configuration mode.

Step 9 **exit**

Example:

```
Router(config-bgp-nbrgrp) # exit
```

Exits the neighbor group configuration mode.

Step 10 **neighbor ip-address**

Example:

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

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 11 **remote-as as-number**

Example:

```
Router(config-bgp-nbr) # remote-as 1
```

Creates a neighbor and assigns a remote autonomous system (AS) number to it.

- Range for 2-byte autonomous system numbers (ASNs) is 1 to 65535.
- Range for 4-byte autonomous system numbers (ASNs) in asplain format is 1 to 4294967295.
- Range for 4-byte autonomous system numbers (ASNs) in asdot format is 1.0 to 65535.65535.

Step 12 **use neighbor-group group-name**

Example:

```
Router(config-bgp-nbr) # use neighbor-group n1
```

Specifies that the BGP neighbor inherit configuration from the specified neighbor group.

Step 13 **address-family ipv4 unicast**

Example:

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

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

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 14 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Disabling Per Neighbor TCP MSS

Perform this task to disable TCP MSS for a particular neighbor under neighbor group.

Procedure

Step 1 **configure****Example:**

```
Router# configure
```

Step 2 **router bgp *as-number*****Example:**

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

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family *ipv4 unicast*****Example:**

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

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 4 **exit****Example:**

```
Router(config-bgp-af)# exit
```

Exits router address family configuration mode, and returns to BGP configuration mode.

Step 5 **neighbor-group *name*****Example:**

```
Router(config-bgp)# neighbor-group n1
```

Enters neighbor group configuration mode.

Step 6 **tcp mss *segment-size*****Example:**

```
Router(config-bgp-nbrgrp)# tcp mss 500
```

Configures TCP maximum segment size. The range is from 68 to 10000.

Step 7 **address-family *ipv4 unicast*****Example:**

```
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
```

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 8 **exit****Example:**

```
Router(config-bgp-nbrgrp-af) # exit
```

Exits router address family configuration mode.

Step 9 **exit****Example:**

```
Router(config-bgp-nbrgrp) # exit
```

Exits the neighbor group configuration mode.

Step 10 **neighbor** *ip-address***Example:**

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

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 11 **remote-as** *as-number***Example:**

```
Router(config-bgp-nbr) # remote-as 1
```

Creates a neighbor and assigns a remote autonomous system (AS) number to it.

- Range for 2-byte autonomous system numbers (ASNs) is 1 to 65535.
- Range for 4-byte autonomous system numbers (ASNs) in asplain format is 1 to 4294967295.
- Range for 4-byte autonomous system numbers (ASNs) in asdot format is 1.0 to 65535.65535.

Step 12 **use neighbor-group** *group-name***Example:**

```
Router(config-bgp-nbr) # use neighbor-group nl
```

Specifies that the BGP neighbor inherit configuration from the specified neighbor group.

Step 13 **tcp mss inheritance-disable****Example:**

```
Router(config-bgp-nbr) # tcp mss inheritance-disable
```

Disables TCP MSS for the neighbor.

Step 14 **address-family** **ipv4 unicast****Example:**

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

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

Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 15 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure BGP Route Filtering by Route Policy

Perform this task to configure BGP routing filtering by route policy.

Procedure

Step 1 **configure**

Step 2 **route-policy** *name*

Example:

```
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
```

(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.

Step 3 **end-policy**

Example:

```
Router(config-rpl)# end-policy
```

(Optional) Ends the definition of a route policy and exits route policy configuration mode.

Step 4 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 5 **neighbor** *ip-address*

Example:

```
Router(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 6 **address-family { ipv4 | ipv6 } unicast**

Example:

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 7 **route-policy *route-policy-name* { in | out }**

Example:

```
Router(config-bgp-nbr-af)# route-policy drop-as-1234 in
```

Applies the specified policy to inbound routes.

Step 8 **commit**

Configure BGP Attribute Filtering

SUMMARY STEPS

1. **configure**
2. **router bgp *as-number***
3. **attribute-filter group *attribute-filter group name***
4. **attribute *attribute code* { discard | treat-as-withdraw }**

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp *as-number***

Example:

```
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 3 **attribute-filter group** *attribute-filter group name*

Example:

```
Router(config-bgp)# attribute-filter group ag_discard_med
```

Specifies the attribute-filter group name and enters the attribute-filter group configuration mode, allowing you to configure a specific attribute filter group for a BGP neighbor.

Step 4 **attribute** *attribute code* { **discard** | **treat-as-withdraw** }

Example:

```
Router(config-bgp-attrfg)# attribute 24 discard
```

Specifies a single or a range of attribute codes and an associated action. The allowed actions are:

- **Treat-as-withdraw**— 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**— Discards this attribute. The matching attributes alone are discarded and the rest of the Update message is processed normally.

Configure BGP Next-Hop Trigger Delay

Perform this task to configure BGP next-hop trigger delay. The Routing Information Base (RIB) classifies the dampening notifications based on the severity of the changes. Event notifications are classified as critical and noncritical. This task allows you to specify the minimum batching interval for the critical and noncritical events.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **nexthop trigger-delay** { **critical** *delay* / **non-critical** *delay* }
5. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family** { **ipv4** | **ipv6** } **unicast**

Example:

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 4 **nexthop trigger-delay** { **critical** *delay* / **non-critical** *delay* }

Example:

```
Router(config-bgp-af)# nexthop trigger-delay critical 15000
```

Sets the critical next-hop trigger delay.

This list provides the default **critical** and **non-critical** delay values for the specified address families.

- **critical** : 3000 msec for all address families, except VPNv4 and VPNv6 address families.
- **critical**: 50 msec for VPNv4 and VPNv6 address families.
- **non-critical**: 10000 msec for all address families.

Avoid configuring the **nexthop trigger-delay critical 0** as it is not suitable on:

- Scaled deployments where a long BGP next-hop walk time duration is inevitable.
- Deployments where BGP next-hop changes are frequent.

Disadvantages of **nexthop trigger-delay critical 0** configuration

- High CPU utilization as each change notification triggers a BGP next-hop walk for address families configured with **nexthop trigger-delay critical 0**.
- BGP next-hop change notifications are not batched. This disallows interleaving of next-hop walks in address families with the non-zero delay configuration as these address families wait until the address families with the zero critical delay value complete their next-hop walks.
- Extended wait time before the BGP next-hop walk starts on address families with the non-zero critical delay configuration, leading to potential traffic blackholing.

Starting with Cisco IOS XR Release 7.10.1, the default critical delay configuration in VPNv4 address family was changed from 0 msec to 50 msec. With this change, all address families have a default non-zero critical delay value. To see the critical delay value of each address family, run the **show bgp all all nexthops** command.

After you have upgraded to Cisco IOS XR Release 7.10.1 or later, if you configure the default critical delay value in the IPv4 address family to 0 msec, you will observe a considerable delay in VPNv4 convergence for the following reasons:

- The IPv4 address families are walked as many times as the number of next-hop critical alerts raised to BGP.

- The BGP next-hop updates for the IPv4 address family prefixes take precedence over VPNv4 address family prefixes.

Advantages of configuring **nexthop trigger-delay critical** with a non-zero default value

- Provides next-hop change notification batching which reduces the number of BGP next-hop walks.
- Allows interleaving different active BGP next-hop walks for the respective address families while prioritizing some address families over the others.

Therefore, we strongly recommend you to configure **nexthop trigger-delay critical** with a non-zero value.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Disable Next-Hop Processing on BGP Updates

Perform this task to disable next-hop calculation for a neighbor and insert your own address in the next-hop field of BGP updates. Disabling the calculation of the best next hop to use when advertising a route causes all routes to be advertised with the network device as the next hop.



Note Next-hop processing can be disabled for address family group, neighbor group, or neighbor address family.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address***Example:**

```
Router(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **remote-as** *as-number***Example:**

```
Router(config-bgp-nbr)# remote-as 206
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 5 **address-family** { **ipv4** | **ipv6** } **unicast****Example:**

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 6 **next-hop-self****Example:**

```
Router(config-bgp-nbr-af)# next-hop-self
```

Sets the next-hop attribute for all routes advertised to the specified neighbor to the address of the local router. Disabling the calculation of the best next hop to use when advertising a route causes all routes to be advertised with the local network device as the next hop.

Step 7 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure BGP Community and Extended-Community Advertisements

Perform this task to specify that community/extended-community attributes should be sent to an eBGP neighbor. These attributes are not sent to an eBGP neighbor by default. By contrast, they are always sent to iBGP neighbors. This section provides examples on how to enable sending community attributes. The **send-community-ebgp** keyword can be replaced by the **send-extended-community-ebgp** keyword to enable sending extended-communities.

If the **send-community-ebgp** command is configured for a neighbor group or address family group, all neighbors using the group inherit the configuration. Configuring the command specifically for a neighbor overrides inherited values.



Note BGP community and extended-community filtering cannot be configured for iBGP neighbors. Communities and extended-communities are always sent to iBGP neighbors under VPNv4, MDT, IPv4, and IPv6 address families.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp *as-number***

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor *ip-address***

Example:

```
Router(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **remote-as *as-number***

Example:

```
Router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 5 **address-family {*ipv4* {*labeled-unicast* | *unicast* | *mdt* | *mvpn* | *rt-filter* | *tunnel*} | *ipv6* {*labeled-unicast* | *mvpn* | *unicast*}}**

Example:

```
Router(config-bgp-nbr)# address-family ipv6 unicast
```

Enters neighbor address family configuration mode for the specified address family. Use either **ipv4** or **ipv6** address family keyword with one of the specified address family sub mode identifiers.

IPv6 address family mode supports these sub modes:

- **labeled-unicast**
- **mvpn**

- **unicast**

IPv4 address family mode supports these sub modes:

- **labeled-unicast**
- **mdt**
- **mvpn**
- **rt-filter**
- **tunnel**
- **unicast**

Step 6 Use one of these commands:

- **send-community-ebgp**
- **send-extended-community-ebgp**

Example:

```
Router(config-bgp-nbr-af) # send-community-ebgp
```

or

```
Router(config-bgp-nbr-af) # send-extended-community-ebgp
```

Specifies that the router send community attributes or extended community attributes (which are disabled by default for eBGP neighbors) to a specified eBGP neighbor.

Step 7 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
 - **No** —Exits the configuration session without committing the configuration changes.
 - **Cancel** —Remains in the configuration session, without committing the configuration changes.
-

Advertising IPv4 NLRI with IPv6 Next Hops in MP-BGP Networks

Table 40: 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 Next Hops 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. These tables 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.

Many multiprotocol network deployments today have topologies of devices configured with one type of address family interspersed with devices configured with a different type of address family. An IPv4 core network can be surrounded by IPv6 devices, and vice versa. Such deployments require the use of multiprotocol BGP (MP-BGP) that allow the advertisement of IPv4 NLRI across IPv6 next hops. Hence, when MP-BGP is

used to advertise the corresponding reachability information in such heterogeneous networks, the BGP router (speaker) advertises the NLRI of a given address family through a next hop of a different address family.

There are several reasons for these heterogeneous deployments. A primary reason is the lack of availability of IPv4 addresses to be used on the interfaces of BGP speakers. A second reason is the intent to move to a pure IPv6 deployment, but in phases. Hence, by configuring the **ipv4 forwarding-enable** command on interfaces, the type of address family used doesn't impact the flow of traffic in the network.

Configure Advertising IPv4 NLRI with IPv6 Next Hops in the Default VRF

This section describes the configuration required to enable IPv4 NLRI advertisement across an IPv6 next hop.

Use the following configuration for advertising IPv4 NLRI through IPv6 next hops.

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.

```
/* Configure the required interfaces with an IPv6 address
   and the ipv4 forwarding-enable command */

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
RP/0/RP0/CPU0:Oct 30 07:20:17.347 UTC: ifmgr[275]: %PKT_INFRA-LINK-3-UPDOWN : Interface
HundredGigE0/0/0/24, changed state to Down
RP/0/RP0/CPU0:ios(config-if)#RP/0/RP0/CPU0:Oct 30 07:20:18.473 UTC: ifmgr[275]:
%PKT_INFRA-LINK-3-UPDOWN : Interface HundredGigE0/0/0/24, changed state to Up
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
Tue Mar  6 10:12:15.948 IST
RP/0/RP0/CPU0:Oct 30 07:22:06.519 UTC: ifmgr[275]: %PKT_INFRA-LINK-3-UPDOWN : Interface
HundredGigE0/0/0/25, changed state to Down
RP/0/RP0/CPU0:ios(config-if)#RP/0/RP0/CPU0:Oct 30 07:22:07.590 UTC: ifmgr[275]:
%PKT_INFRA-LINK-3-UPDOWN : Interface HundredGigE0/0/0/25, changed state to Up
Router(config-if)# exit

/* Create a route policy to set an IPv6 nexthop for IPv4 routes */
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
Fri Oct 30 07:30:58.627 UTC

/* Configure an iBGP peer through the HundredGigE0/0/0/25 interface */
Router(config)# route-policy pass-all
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config)# router bgp 100
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
Tue Mar  6 10:16:07.134 IST
Router(config-bgp-nbr-af)# root

```

```

/* 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 */
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
Tue Mar  6 10:21:19.434 IST
Router(config-bgp-nbr-af)# root

```

Running Configuration

```

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

```

Verification

Use the following show commands to verify the advertisement of IPv4 NLRI through IPv6 nexthops.

```

/* Verify BGP neighbor configuration and the advertisement of nexthops */
RRouter# 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: 0x60000001, Version: 0x2
  Nexthop Flags: 0x00000080
  Nexthop Handle: 0x7f53e85b136c

RIB Related Information:
Firsthop interface handle 0x000000140
  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 Nexthop 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]
  Metrics: [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 Advertising IPv4 NLRI with IPv6 Next Hops in the Non-Default VRF

This section describes the configuration required to enable IPv4 NLRI advertisement across an IPv6 next hop in the non-default VRF.

Use the following configuration for advertising IPv4 NLRI through IPv6 nexthops.

In the following example, the eBGP peer is configured on the GigabitEthernet 0/1/0/0.2 interface, and the iBGP peer is configured on the GigabitEthernet 0/1/0/1.2 interface of the router being configured.

Procedure

Step 1 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

```

Step 8 Perform the following steps to verify the configuration that you have configured:

- a) 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
```

- b) Verify update groups for the eBGP and iBGP neighbours.

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
```

- c) 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

```

d) 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)

```

e) 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

```

f) Verify IPv6 nexthops for the eBGP and iBGP routes.

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
```

```
                PR Pending Registration
```

```
                I Invalid (Policy drop)
```

Next Hop	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

g) 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 Nexthop 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:    0
  Most Recent Events:
    Time                Event Type      Metric
    Mar 27 23:06:16.748  Reachable          0

Prefix Related Information
  Active Tables: [IPv4 Unicast]
  Metrics: [0xffffffff]
  Reference Counts: [1]
  Encapsulations: []
  Interface Handle: 0x0

Linked Nexthop Count: 1
Attr ref-count: 3

```

h) Verify IPv6 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 Nexthop 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: 0
Most Recent Events:
  Time                Event Type      Metric
  Mar 27 23:06:16.748 Reachable      0

Prefix Related Information
Active Tables: [IPv4 Unicast]
Metricres: [0xffffffff]
Reference Counts: [1]
Encapsulations: []
Interface Handle: 0x0
Attr ref-count: 4

```

- i) Verify eBGP and iBGP routes in RIB.

Example:

```

Router# show route vrf RFC5549 192.1.0.1/32
Wed Mar 27 23:46:00.095 IST

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.

```

- j) Verify eBGP and iBGP routes in RIB with the *detail* option.

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.

```

k) Verify eBGP and iBGP routes in CEF.

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

```

- l) Verify eBGP and iBGP routes in CEF with the *detail* option.

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

```

Configure BGP Cost Community

BGP receives multiple paths to the same destination and it uses the best-path algorithm to decide which is the best path to install in RIB. To enable users to determine an exit point after partial comparison, the cost community is defined to tie-break equal paths during the best-path selection process. Perform this task to configure the BGP cost community.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **route-policy name**

Example:

```
Router(config)# route-policy costA
```

Enters route policy configuration mode and specifies the name of the route policy to be configured.

Step 3 **set extcommunity cost { cost-extcommunity-set-name | cost-inline-extcommunity-set } [additive]**

Example:

```
Router(config)# set extcommunity cost cost_A
```

Specifies the BGP extended community attribute for cost.

Step 4 **end-policy**

Example:

```
Router(config)# end-policy
```

Ends the definition of a route policy and exits route policy configuration mode.

Step 5 `router bgp as-number`

Example:

```
Router(config)# router bgp 120
```

Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 6 Do one of the following:

- **default-information originate**
- **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]

Applies the cost community to the attach point (route policy).

Step 7 Do one of the following:

- **neighbor** *ip-address* **remote-as** *as-number*
- **route-policy** *route-policy-name* { **in** | **out** }

Step 8 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 9 `show bgp ip-address`

Example:

```
Router# show bgp 172.168.40.24
```

Displays the cost community in the following format:

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

Configure Software to Store Updates from Neighbor

Perform this task to configure the software to store updates received from a neighbor.

The **soft-reconfiguration inbound** command causes a route refresh request to be sent to the neighbor if the neighbor is route refresh capable. If the neighbor is not route refresh capable, the neighbor must be reset to relearn received routes using the **clear bgp soft** command.



Note Storing updates from a neighbor works only if either the neighbor is route refresh capable or the **soft-reconfiguration inbound** command is configured. Even if the neighbor is route refresh capable and the **soft-reconfiguration inbound** command is configured, the original routes are not stored unless the **always** option is used with the command. The original routes can be easily retrieved with a route refresh request. Route refresh sends a request to the peer to resend its routing information. The **soft-reconfiguration inbound** command stores all paths received from the peer in an unmodified form and refers to these stored paths during the clear. Soft reconfiguration is memory intensive.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **address-family** { **ipv4** | **ipv6** } **unicast**
5. **soft-reconfiguration inbound** [**always**]
6. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
Enters mode.
```

Step 2 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address*

Example:

```
Router(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **address-family** { **ipv4** | **ipv6** } **unicast**

Example:

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 5 **soft-reconfiguration inbound [always]**

Example:

```
Router(config-bgp-nbr-af)# soft-reconfiguration inbound always
```

Configures the software to store updates received from a specified neighbor. Soft reconfiguration inbound causes the software to store the original unmodified route in addition to a route that is modified or filtered. This allows a “soft clear” to be performed after the inbound policy is changed.

Soft reconfiguration enables the software to store the incoming updates before apply policy if route refresh is not supported by the peer (otherwise a copy of the update is not stored). The **always** keyword forces the software to store a copy even when route refresh is supported by the peer.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

BGP Persistence

BGP persistence enables the local router to retain routes that it has learnt from the configured neighbor even after the neighbor session is down. BGP persistence is also referred as Long Lived Graceful Restart (LLGR). LLGR takes effect after graceful restart (GR) ends or immediately if GR is not enabled. LLGR ends either when the LLGR stale timer expires or when the neighbor sends the end-of-RIB marker after it has revised its routes. When LLGR for a neighbor ends, all routes from that neighbor that are still stale will be deleted. The LLGR capability is signaled to a neighbor in the BGP OPEN message if it has been configured for that neighbor. LLGR differs from graceful restart in the following ways.

- It can be in effect for a much longer time than GR
- LLGR stale routes are least preferred during route selection (bestpath computation).
- An LLGR stale route will be advertised with the LLGR_STALE community attached if it is selected as best path. It will not be advertised at all to routers that are not LLGR capable.
- LLGR stale routes will not be deleted when the forwarding path to the neighbor is detected to be down.
- An LLGR stale route will not be deleted if the BGP session to the neighbor goes down multiple times even if that neighbor does not re-advertise the route.
- Any route that has the NO_LLGR community will not be retained.



Note You can disable GR helper-only for peer-group and neighbor, when there is no global GR helper-only configured.

BGP will not pass the updates containing communities 65535:6, 65535:7 to its neighbors until the neighbors negotiate BGP persistence capabilities. The communities 65535:6 and 65535:7 are reserved for LLGR_STALE and NO_LLGR respectively, BGP behavior maybe unpredictable if you have configured these communities prior to release 5.2.2. We recommend not to configure the communities 65535:6 and 65535:7.

The BGP persistence feature is supported only on the following AFIs:

- VPNv4 and VPNv6
- RT constraint
- Flow spec (IPv4, IPv6, VPNv4 and VPNv6)
- IPv4 and IPv6 address family

BGP Persistence Configuration: Example

This example sets long lived graceful restart (LLGR) stale-time of 16777215 on BGP neighbor 10.3.3.3.

```
router bgp 100
 neighbor 10.3.3.3
   remote-as 30813
   update-source Loopback0
   graceful-restart stalepath-time 150
   address-family vpnv4 unicast
     long-lived-graceful-restart capable
     long-lived-graceful-restart stale-time send 16777215 accept 16777215
   !
   address-family vpnv6 unicast
     long-lived-graceful-restart capable
     long-lived-graceful-restart stale-time send 16777215 accept 16777215
```

Flexible BGP Persistence

Table 41: Feature History Table

Feature Name	Release Name	Description
Flexible BGP Persistence	Release 24.3.1	

Feature Name	Release Name	Description
		<p>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])</p> <p>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.</p> <p>The feature introduces these changes:</p> <p>CLI:</p> <ul style="list-style-type: none"> • The default, any, and advertise-internal-only keywords are added to the long-lived-graceful-restart command. • The fields Long-lived Graceful Restart Stale Time Send Default, Default 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. <p>YANG Data Model:</p> <ul style="list-style-type: none"> • Cisco-IOS-XR-ipv4-bgp-cfg (see GitHub, YANG Data Models Navigator)

Flexible BGP Persistence and its Benefits

The Flexible BGP Persistence feature provides you the flexibility to enable LLGR without manual timeout configuration, advertise LLGR stale routes to non-LLGR eBGP neighbors seamlessly, and ensures controlled route distribution and policy enforcement by attaching NO_EXPORT community and setting local preference to 0.

- **Simplified configuration:** You can configure the router to enable and advertise the LLGR capability without explicitly setting a timeout value, and to accept the LLGR stale time provided by the peer without imposing its own threshold, by using the **long-lived-graceful-restart send default accept any** command.
- The **send default** keyword pair provides the flexibility from not having to specify a timeout value.

The **accept any** keyword pair allows the router to accept the LLGR stale time provided by the peer without imposing its own threshold for acceptance. This flexibility ensures compatibility with a wider range of peer configurations and simplifies the management of LLGR settings.

- **Enhanced network resilience:** This feature allows LLGR stale routes to be advertised to non-LLGR capable eBGP neighbors, enhancing network resilience. This capability ensures that even non-LLGR capable neighbors can benefit from the stability provided by LLGR, thereby contributing to overall network robustness.
- **Enhanced network stability:** This feature allows the attachment of the NO_EXPORT community and setting the local preference to 0 when advertising an LLGR route to an internal neighbor that has not negotiated the capability. The LLGR_STALE community is always sent along with NO_EXPORT community. The feature allows explicit configuration to enable this requirement through the **long-lived-graceful-restart capable advertise-internal-only** command.

This approach prevents the stale route from being propagated beyond the local Autonomous System (AS) and ensures that it is not preferred over other routes. This helps maintain network stability and prevents potential routing issues by ensuring that only capable and properly negotiated routes are used for forwarding traffic.

Configure Flexible BGP Persistence

Configure LLGR Advertisement and Activation with Default and Peer-Specified Stale Time

Procedure

- Step 1** Configure LLGR advertisement and activation with default and peer-specified stale time.

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
```

Configure the local router to advertise the LLGR capability to the neighbor and activate LLGR for prefixes received from the neighbor, specify that the router sends the default value, which is 172800 seconds, for the LLGR time, and ensure that the router accepts the LLGR stale time provided by the peer without imposing its own acceptance threshold.

Step 2 Verify that the router is configured to send a default LLGR stale time to its BGP neighbors, that the default stale time being advertised is 172800 seconds, and that the router accepts any LLGR stale time value provided by the peer without imposing its own threshold.

Example:

```
Router(config)# show bgp neighbor 192.0.2.254
...
AF-dependent capabilities:
  Long-lived Graceful Restart Capability advertised
    Advertised Long-lived Stale time 172800 seconds
  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
  Maximum acceptable long-lived stale time from this neighbor is 16777215
  Long-lived Graceful Restart Capability received
    Received long-lived stale time is 172800 seconds
    Neighbor preserved the forwarding state during latest restart
...
```

Here is an explanation of the highlighted fields:

- **Long-lived Graceful Restart Stale Time Send Default is ON:** This field indicates that the router is configured to send a default value for the LLGR stale time to its BGP neighbors. The default stale time is used when the router advertises its LLGR capability.
- **Default advertised long-lived stale time is 172800 seconds:** This field specifies the default LLGR stale time value that the router advertises to its BGP neighbors. In this case, Cisco sets the default stale time to 172800 seconds, which is equivalent to 2 days, in the BGP configuration parameters.
- **Long-lived Graceful Restart Stale Time Accept Any is ON:** This field indicates that the router is configured to accept any LLGR stale time value provided by its BGP neighbors. The router does not impose its own threshold for the stale time and accepts the value sent by the peer.

Enable the LLGR Capability and Advertise It Only to iBGP Peers

Procedure

Step 1 Enable the 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
```

Configure the router to enable the LLGR capability and advertise it exclusively to iBGP peers. Set the local preference of the router to 0 to ensure it is not preferred for outbound traffic within the AS and is utilized only when no other routes are available. Set the community attribute to llgr-stale to retain the route temporarily during a graceful restart event. Additionally, set the community attribute to no-export to limit the propagation of the route within the local AS.

Step 2 Verify 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
```

BGP Graceful Maintenance

When a BGP link or router is taken down, other routers in the network find alternative paths for the traffic that was flowing through the failed router or link, if such alternative paths exist. The time required before all routers involved can reach a consensus about an alternate path is called convergence time. During convergence time, traffic that is directed to the router or link that is down is dropped. The BGP Graceful Maintenance feature allows the network to perform convergence before the router or link is taken out of service. The router or link remains in service while the network reroutes traffic to alternative paths. Any traffic that is yet on its way to the affected router or link is still delivered as before. After all traffic has been rerouted, the router or link can safely be taken out of service.

The Graceful Maintenance feature is helpful when alternate paths exist and these alternate paths are not known to routers at the time that the primary paths are withdrawn. The feature provides these alternate paths before the primary paths are withdrawn. The feature is most helpful in networks where convergence time is long. Several factors, such as large routing tables and presence of route reflectors, can result in longer convergence time.

When a BGP router or link is brought into service, the possibility of traffic loss during convergence also exists, although it is less than when a router or link is taken out of service. The BGP Graceful Maintenance feature can also be used in this scenario.

Restrictions for BGP Graceful Maintenance

The following restrictions apply for BGP Graceful Maintenance:

- If the affected router is configured to send the GSHUT community attribute, then other routers in the network that receive it must be configured to interpret it. You must match the community with a routing policy and set a lower preference.
- The LOCAL_PREF attribute is not sent to another AS. Therefore, the LOCAL_PREF option cannot be used on an eBGP link.

**Note**

This restriction does not apply to eBGP links between member-ASs of an AS confederation.

- Alternative routes must exist in the network, otherwise advertising a lower preference has no effect. For example, there is no advantage in configuring Graceful Maintenance for a singly-homed customer router which does not have alternate routes.

- If time consuming policies exist, either at the output of the sending router or at the input of the receiving router, the Graceful Maintenance operation can take a long time.
- Configuring an eBGP ASBR neighbor results in advertising an implicit null label for directly connected routes via BGP. If a user shuts down an eBGP neighbor, the label is not reprogrammed as the system withdraws rewrites on any neighbor state changes. Implicit null label feature support helps avoid churn in terms of adding or removing rewrites for neighbor flaps.

Graceful Maintenance Operation

When Graceful Maintenance is activated, the affected routes are advertised again with a reduced preference. This causes neighboring routers to choose alternative routes. You can use any of the following methods to signal reduced route preference:

- **Add GSHUT community:** Use this method to allow remote routers the freedom to set a preference. Receiving routers must match this community in a policy and set their own preference.
- **Reduce LOCAL_PREF value:** This works for internal BGP neighbors. Use this method if remote routers do not match the GSHUT community.
- **Prepend AS Path:** This works for both internal and external BGP neighbors. Use this method if remote routers do not match the GSHUT community.

When Graceful Maintenance is activated on a BGP connection, the following two operations happen:

1. All routes received from the connection are re-advertised to other neighbors with a lower preference. Note, this happens to only those routes that have actually been advertised to other neighbors. It is possible that a received route was not selected as the best path and therefore not advertised. In that case, it will not be re-advertised.
2. All routes that were advertised to the connection is re-advertised with a lower preference.

In order for the first operation to happen, all routes received from the connection are tagged with an internal attribute called graceful-shut. This attribute is stored internal to only the router; it is not advertised by BGP. This attribute can be seen when the route is displayed with the **show bgp** command. It is different from the GSHUT community. The GSHUT community is advertised by BGP and can be seen in the community list when the route is displayed with the **show bgp** command.

All routes that have the graceful-shut attribute are given the lowest preference during route-selection. Any new route updates that are sent or received on a BGP session under Graceful Maintenance are also treated as described above.

Inter Autonomous System

Advertising a lower preference to another AS in the public Internet may cause unnecessary routing advertisements in distant networks, which may not be desirable. An additional configuration under the neighbor address family, **send-community-gshut-ebgp**, is necessary for the router to originate the GSHUT community to the eBGP neighbor.



Note This does not affect the GSHUT community on a route that already had this community when it was received; it only affects the GSHUT community when this router adds it.

When to Shut Down After Graceful Maintenance

The router or link can be shut down after the network has converged as a result of a graceful-maintenance activation. Convergence can take from less than a second to more than an hour. Unfortunately, a single router cannot know when a whole network has converged. After a graceful-maintenance activation, it can take a few seconds to start sending updates. Then, the “InQ” and “OutQ” of neighbors in the **show bgp <vrf> <afi> <safi> summary** command's output indicates the level of BGP messaging. Both InQ and OutQ should be 0 after convergence. Neighbors should stop sending traffic. However, they won't stop sending traffic if they do not have alternate paths; and in that case traffic loss cannot be prevented.

Activate Graceful Maintenance under BGP Router (All Neighbors)

Activating Graceful Maintenance under a BGP router results in **activate** being configured under **graceful-maintenance** for all neighbors. With just this one configuration, you get the same result if you were to go to every neighbor that has **graceful-maintenance** configured, and added **activate** under it. If you add the keyword **all-neighbors**, thus, **graceful-maintenance activate all-neighbors**, then the router acts as if you configured **graceful-maintenance activate** under every neighbor.



Note We suggest that you activate Graceful Maintenance under a BGP router instance only if it is acceptable to send the GSHUT community for all routes on every neighbor. Re-sending all routes to every neighbor can take significant amount of time on a large router. Sending GSHUT to a neighbor that does not have alternative routes is pointless. If a router has many of such neighbors then a significant amount of time can be saved by not activating Graceful Maintenance on them.

The BGP Graceful Maintenance feature allows you to enable Graceful Maintenance either on a single neighbor, on a group of neighbors across BGP sessions, or on all neighbors. Enabling Graceful Maintenance under a neighbor sub-mode, does two things:

1. All routes that are advertised to this neighbor that has the graceful-shut attribute are advertised to that neighbor with the GSHUT community.
2. Enters graceful-maintenance configuration mode to allow further configuration.

Using the **activate** keyword under graceful-maintenance, causes the following:

1. All routes that are received from this neighbor acquire the graceful-shut attribute.
2. All routes that are advertised to this neighbor are re-advertised to that neighbor with the GSHUT community.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*

3. **graceful-maintenance activate** [**all-neighbors** | **retain-routes**]
4. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure Example: <pre>RP/0/RP0/CPU0:router# configure</pre>	Enters mode.
Step 2	router bgp <i>as-number</i> Example: <pre>Router(config)# router bgp 120</pre>	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	graceful-maintenance activate [all-neighbors retain-routes] Example: <pre>Router(config-bgp)# graceful-maintenance activate all-neighbors</pre>	<p>Announces routes with the g-shut community and other attributes as configured under the neighbors. This causes neighbors to reject routes from this router and choose alternates. This allows the router to be gracefully brought in or out of service.</p> <p>If you use the all-neighbors keyword, Graceful Maintenance is activated even for those neighbors that do not have it activated. Choosing retain-routes causes RIB to retain BGP routes when the BGP process is stopped.</p> <p>Use the retain-routes option when only BGP must be brought down instead of the entire router, and when it is known that neighboring routers are kept in operation during the maintenance of the local BGP. If RIB has alternative routes provided by another protocol or a default route, then it is recommended that you do not to retain BGP routes after the BGP process stops.</p>
Step 4	Use the commit or end command.	<p>commit —Saves the configuration changes and remains within the configuration session.</p> <p>end —Prompts user to take one of these actions:</p> <ul style="list-style-type: none"> • Yes — Saves configuration changes and exits the configuration session. • No —Exits the configuration session without committing the configuration changes. • Cancel —Remains in the configuration session, without committing the configuration changes.

What to do next

After activating Graceful Maintenance, you must wait for all the routes to be sent and for the neighboring routers to redirect their traffic away from the router or link under maintenance. After the traffic is redirected, then it is safe to take the router or link out of service. While there is no definitive way to know when all the routes have been sent, you can use the **show bgp summary** command to check the OutQ of the neighbors. When OutQ reaches a value 0, there are no more updates to be sent.

Activate Graceful Maintenance on a Single Neighbor

Use the following steps to activate Graceful Maintenance for a single neighbor:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **graceful-maintenance activate**
5. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure Example: RP/0/RP0/CPU0:router# configure	Enters mode.
Step 2	router bgp <i>as-number</i> Example: Router(config)# router bgp 120	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	neighbor <i>ip-address</i> Example: Router(config-bgp)# neighbor 172.168.40.24	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	graceful-maintenance activate Example: Router(config-bgp-nbr)# graceful-maintenance activate	Announces routes with Graceful Maintenance attributes.
Step 5	Use the commit or end command.	commit —Saves the configuration changes and remains within the configuration session. end —Prompts user to take one of these actions:

	Command or Action	Purpose
		<ul style="list-style-type: none"> • Yes — Saves configuration changes and exits the configuration session. • No — Exits the configuration session without committing the configuration changes. • Cancel — Remains in the configuration session, without committing the configuration changes.

Activate Graceful Maintenance on a Group of Neighbors

Use the following steps to activate Graceful Maintenance on a group of neighbors:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor-group** *Neighbor-group name*
4. **graceful-maintenance activate**
5. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure Example: RP/0/RP0/CPU0:router# configure	Enters mode.
Step 2	router bgp <i>as-number</i> Example: Router(config)# router bgp 120	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	neighbor-group <i>Neighbor-group name</i> Example: Router(config-bgp)# neighbor-group AS_1	Places the router in neighbor group configuration mode.
Step 4	graceful-maintenance activate Example: Router(config-bgp-nbrgrp)# graceful-maintenance activate	Announces routes with Graceful Maintenance attributes.

	Command or Action	Purpose
Step 5	Use the commit or end command.	<p>commit —Saves the configuration changes and remains within the configuration session.</p> <p>end —Prompts user to take one of these actions:</p> <ul style="list-style-type: none"> • Yes — Saves configuration changes and exits the configuration session. • No —Exits the configuration session without committing the configuration changes. • Cancel —Remains in the configuration session, without committing the configuration changes.

What to do next

You must configure the **send-community-gshut-ebgp** command under the neighbor address family of an eBGP neighbor for this router to add the GSHUT community.



Note Sending GSHUT community may not be desirable under every address family of an eBGP neighbor. To allow you to target GSHUT community to a specific set of address families, use the **send-community-gshut-ebgp** command.

Direct Router to Reduce Route Preference

The BGP Graceful Maintenance feature works only with the availability of alternate paths. You must advertise routes with a lower preference to allow alternate routes to take over before taking down a link or router. Use the following steps to modify the route preference:



Note Attributes for graceful maintenance are added to a route update message after an outbound policy has been applied to it.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 120
```

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address*

Example:

```
Router(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **remote-as** *as-number*

Example:

```
Router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 5 **graceful-maintenance as-prepends** *value* | **local-preference** *value*

Example:

```
Router(config-bgp-nbr)# graceful-maintenance  
local-preference 4
```

Specifies the number of times the local AS number is to be prepended to the AS path of routes and advertises the GSHUT community with the local preference value specified for the routes. When the router adds the GSHUT community to a route as it advertises it, it also changes the LOCAL_PREF attribute and prepends the local AS number as specified in the commands. Sending GSHUT provides flexibility in the manner in which neighboring routers handle the lower preference: they can match it in a route policy and do the most appropriate thing with it. On the other hand, in simple networks, it is easier to set local-preference to 0, than to create route policies everywhere else.

Note

LOCAL_PREF is not sent to real eBGP neighbors, but sent to confederation member AS eBGP neighbors. To lower the preference to eBGP neighbors, as-prepends value is required.

Example: Configure route policy matching GSHUT community to lower route preference

```
route-policy gshut  
  if community matches-any gshut then  
    set local-preference 0  
  endif  
  pass  
end-policy
```

```
neighbor 666.0.0.3  
  address-family ipv4 unicast  
    route-policy gshut in
```



Note 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 is taken out of maintenance, the attribute on its paths is removed, but not the community. The attribute is internal and not sent in BGP messages. It is used to reject routes during path selection.

Bring Router or Link Back into Service

Before you bring the router or link back into service, you must first activate graceful maintenance and then remove the **activate** configuration.

Show Command Outputs to Verify BGP Graceful Maintenance

This section lists the show commands you can use to verify that BGP Graceful Maintenance is activated and check related attributes:

Use the **show bgp <IP address>** command to display graceful-shutdown community and the graceful-shut path attribute with BGP graceful maintenance activated:

```
RP/0/0/CPU0:R4#show bgp 5.5.5.5
...
10.10.10.1 from 10.10.10.1 (192.168.0.5)
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: 192.168.0.5, Cluster list: 192.168.0.1
```

The following is sample output from the **show bgp community graceful-shutdown** command displaying the graceful maintenance feature information:

```
RP/0/0/CPU0:R4#show bgp community graceful-shutdown
BGP router identifier 192.168.0.4, 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
* 5.5.5.5/32 10.10.10.1 88 0 1 ?
Processed 1 prefixes, 1 paths
```

The following is the sample output from the **show bgp neighbors** command with the ip-address and configuration argument and keyword to display graceful maintenance feature attributes:

```
RP/0/0/CPU0:R1#show bgp neighbor 12.12.12.5
...
Graceful Maintenance locally active, Local Pref=45, AS prepends=3
...
For Address Family: IPv4 Unicast
```

```

...
GSHUT Community attribute sent to this neighbor
...
*****
RP/0/0/CPU0:R1#show bgp neighbor 12.12.12.5 configuration
neighbor 12.12.12.5
remote-as 1 []
graceful-maintenance 1 []
gr-maint local-preference 45 []
gr-maint as-prepends 3 []
gr-maint activate []

```

The following is the sample output of the **show rpl community-set** command with graceful maintenance feature attributes displayed:

```

RP/0/0/CPU0:R5#show rpl community-set
Listing for all Community Set objects
community-set gshut
graceful-shutdown
end-set

```

The following is the sample of the syslog that is issued when a BGP neighbor that has graceful maintenance activated, comes up. It is a warning text that reminds you to deactivate graceful maintenance after convergence.

```

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

```

Bring Router or Link Back into Service

Before you bring the router or link back into service, you must first activate graceful maintenance and then remove the **activate** configuration.

Show Command Outputs to Verify BGP Graceful Maintenance

This section lists the show commands you can use to verify that BGP Graceful Maintenance is activated and check related attributes:

Use the **show bgp <IP address>** command to display graceful-shutdown community and the graceful-shut path attribute with BGP graceful maintenance activated:

```

RP/0/0/CPU0:R4#show bgp 5.5.5.5
...
10.10.10.1 from 10.10.10.1 (192.168.0.5)
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: 192.168.0.5, Cluster list: 192.168.0.1

```

The following is sample output from the **show bgp community graceful-shutdown** command displaying the graceful maintenance feature information:

```

RP/0/0/CPU0:R4#show bgp community graceful-shutdown
BGP router identifier 192.168.0.4, 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
* 5.5.5.5/32 10.10.10.1 88 0 1 ?
Processed 1 prefixes, 1 paths

```

The following is the sample output from the **show bgp neighbors** command with the ip-address and configuration argument and keyword to display graceful maintenance feature attributes:

```

RP/0/0/CPU0:R1#show bgp neighbor 12.12.12.5
...
Graceful Maintenance locally active, Local Pref=45, AS prepends=3
...
For Address Family: IPv4 Unicast
...
GSHUT Community attribute sent to this neighbor
...
*****
RP/0/0/CPU0:R1#show bgp neighbor 12.12.12.5 configuration
neighbor 12.12.12.5
remote-as 1 []
graceful-maintenance 1 []
gr-maint local-preference 45 []
gr-maint as-prepend 3 []
gr-maint activate []

```

The following is the sample output of the **show rpl community-set** command with graceful maintenance feature attributes displayed:

```

RP/0/0/CPU0:R5#show rpl community-set
Listing for all Community Set objects
community-set gshut
graceful-shutdown
end-set

```

The following is the sample of the syslog that is issued when a BGP neighbor that has graceful maintenance activated, comes up. It is a warning text that reminds you to deactivate graceful maintenance after convergence.

```

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

```

Flow-tag propagation

The flow-tag propagation feature enables you to establish a co-relation between route-policies and user-policies. Flow-tag propagation using BGP allows user-side traffic-steering based on routing attributes such as, AS number, prefix lists, community strings and extended communities. Flow-tag is a logical numeric identifier that is distributed through RIB as one of the routing attribute of FIB entry in the FIB lookup table. A flow-tag is instantiated using the 'set' operation from RPL and is referenced in the C3PL PBR policy, where it is associated with actions (policy-rules) against the flow-tag value.

You can use flow-tag propagation to:

- Classify traffic based on destination IP addresses (using the Community number) or based on prefixes (using Community number or AS number).
- Select a TE-group that matches the cost of the path to reach a service-edge based on customer site service level agreements (SLA).

- Apply traffic policy (TE-group selection) for specific customers based on SLA with its clients.
- Divert traffic to application or cache server.

Restrictions for Flow-Tag Propagation

Some restrictions are placed with regard to using Quality-of-service Policy Propagation Using Border Gateway Protocol (QPPB) and flow-tag feature together. These include:

- A route-policy can have either 'set qos-group' or 'set flow-tag,' but not both for a prefix-set.
- Route policy for qos-group and route policy flow-tag cannot have overlapping routes. The QPPB and flow tag features can coexist (on same as well as on different interfaces) as long as the route policy used by them do not have any overlapping route.
- Mixing usage of qos-group and flow-tag in route-policy and policy-map is not recommended.

Source and destination-based flow tag

The source-based flow tag feature allows you to match packets based on the flow-tag assigned to the source address of the incoming packets. Once matched, you can then apply any supported PBR action on this policy.

Configure Source and Destination-based Flow Tag

This task applies flow-tag to a specified interface. The packets are matched based on the flow-tag assigned to the source address of the incoming packets.



Note You will not be able to enable both QPPB and flow tag feature simultaneously on an interface.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **ipv4 | ipv6 bgp policy propagation input flow-tag {destination | source}**
4. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
Enters mode.
```

Step 2 **interface** *type interface-path-id*

Example:

```
Router(config-if)# interface FourHundredGige 0/1/0/0
```

Enters interface configuration mode and associates one or more interfaces to the VRF.

Step 3 **ipv4 | ipv6 bgp policy propagation input flow-tag {destination | source}****Example:**

```
Router(config-if)# ipv4 bgp policy propagation input flow-tag source
```

Enables flow-tag policy propagation on source or destination IP address on an interface.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes, and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration mode, without committing the configuration changes.

Example

The following show commands display outputs with PBR policy applied on the router:

```
show running-config interface gigabitEthernet 0/0/0/12
Thu Feb 12 01:51:37.820 UTC
interface GigabitEthernet0/0/0/12
 service-policy type pbr input flowMatchPolicy
 ipv4 bgp policy propagation input flow-tag source
 ipv4 address 192.5.1.2 255.255.255.0
!
```

```
Router#show running-config policy-map type pbr flowMatchPolicy
Thu Feb 12 01:51:45.776 UTC
policy-map type pbr flowMatchPolicy
 class type traffic flowMatch36
   transmit
 !
 class type traffic flowMatch38
   transmit
 !
 class type traffic class-default
 !
 end-policy-map
!
```

```
Router#show running-config class-map type traffic flowMatch36
Thu Feb 12 01:52:04.838 UTC
class-map type traffic match-any flowMatch36
 match flow-tag 36
 end-class-map
!
```


Configure Keychains for BGP

Keychains provide secure authentication by supporting different MAC authentication algorithms and provide graceful key rollover. Perform this task to configure keychains for BGP. This task is optional.

**Note**

If a keychain is configured for a neighbor group or a session group, a neighbor using the group inherits the keychain. Values of commands configured specifically for a neighbor override inherited values.

Procedure

Step 1 **configure****Example:**

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number***Example:**

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address***Example:**

```
Router(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **remote-as** *as-number***Example:**

```
Router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 5 **keychain** *name***Example:**

```
Router(config-bgp-nbr)# keychain kych_a
```

Configures keychain-based authentication.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.

- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring an MDT Address Family Session in BGP

Perform this task to configure an IPv4 multicast distribution tree (MDT) subaddress family identifier (SAFI) session in BGP, which can also be used for MVPNv6 network distribution.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **exit**
5. **address-family** { **vpn4** | **vpn6** } **unicast**
6. **exit**
7. **address-family** **ipv4 mdt**
8. **exit**
9. **neighbor** *ip-address*
10. **remote-as** *as-number*
11. **update-source** *interface-type interface-id*
12. **address-family** { **ipv4** | **ipv6** } **unicast**
13. **exit**
14. **address-family** { **vpn4** | **vpn6** } **unicast**
15. **exit**
16. **address-family** **ipv4 mdt**
17. **exit**
18. **vrf** *vrf-name*
19. **rd** { *as-number:nn* | *ip-address:nn* | **auto** }
20. **address-family** { **ipv4** | **ipv6** } **unicast**
21. Do one of the following:
 - **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
 - **redistribute eigrp** *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
 - **redistribute isis** *process-id* [**level** { **1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
 - **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
 - **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
 - **redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
 - **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
22. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure Example: RP/0/RP0/CPU0:router# configure	Enters mode.
Step 2	router bgp <i>as-number</i> Example: Router(config)# router bgp 120	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	address-family { ipv4 ipv6 } unicast Example: Router(config-vrf)# address-family ipv4 unicast	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submenu. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	exit Example: Router(config-bgp-af)# exit	Exits the current configuration mode.
Step 5	address-family { vpnv4 vpnv6 } unicast Example: Router(config-bgp)# address-family vpnv4 unicast	Specifies the address family and enters the address family configuration submenu. To see a list of all the possible keywords and arguments for this command, use the CLI help (?). Note Required if you are configuring multicast MVPN. If configuring MVPNv6, use the vpnv6 keyword
Step 6	exit Example: Router(config-bgp-af)# exit	Exits the current configuration mode.
Step 7	address-family ipv4 mdt Example: Router(config-bgp)# address-family ipv4 mdt	Specifies the multicast distribution tree (MDT) address family.
Step 8	exit Example: Router(config-bgp-af)# exit	Exits the current configuration mode.

	Command or Action	Purpose
Step 9	neighbor <i>ip-address</i> Example: Router(config-bgp)# neighbor 172.168.40.24	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 10	remote-as <i>as-number</i> Example: Router(config-bgp-nbr)# remote-as 2002	Creates a neighbor and assigns a remote autonomous system number to it.
Step 11	update-source <i>interface-type interface-id</i> Example: Router(config-bgp-nbr)# update-source loopback 0	Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor. The <i>interface-type interface-id</i> arguments specify the type and ID number of the interface, such as ATM, POS, Loopback. Use the CLI help (?) to see a list of all the possible interface types and their ID numbers.
Step 12	address-family { ipv4 ipv6 } unicast Example: Router(config-vrf)# address-family ipv4 unicast	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 13	exit Example: Router(config-bgp-nbr-af)# exit	(Optional) Exits the current configuration mode.
Step 14	address-family { vpnv4 vpnv6 } unicast Example: Router(config-bgp-nbr)# address-family vpnv4 unicast	(Optional) Enters address family configuration submode for the specified address family. Note Required if you are configuring multicast MVPN. If configuring MVPNv6, use the vpnv6 keyword.
Step 15	exit Example: Router(config-bgp-nbr-af)# exit	Exits the current configuration mode.
Step 16	address-family ipv4 mdt Example: Router(config-bgp)# address-family ipv4 mdt	Specifies the multicast distribution tree (MDT) address family.
Step 17	exit Example: Router(config-bgp-af)# exit	Exits the current configuration mode.
Step 18	vrf <i>vrf-name</i> Example:	(Optional) Enables BGP routing for a particular VRF on the PE router.

	Command or Action	Purpose
	Router(config-bgp)# vrf vpn1	Note Required if you are configuring multicast MVPN.
Step 19	rd { as-number:nn ip-address:nn auto } Example: Router(config-bgp-vrf)# rd 1:1	(Optional) Configures the route distinguisher. <ul style="list-style-type: none"> Use the auto keyword if you want the router to automatically assign a unique RD to the VRF. Automatic assignment of RDs is possible only if a router ID is configured using the bgp router-id command in router configuration mode. This allows you to configure a globally unique router ID that can be used for automatic RD generation. <p>The router ID for the VRF does not need to be globally unique, and using the VRF router ID would be incorrect for automatic RD generation. Having a single router ID also helps in checkpointing RD information for BGP graceful restart, because it is expected to be stable across reboots.</p> Note Required if you are configuring multicast MVPN.
Step 20	address-family { ipv4 ipv6 } unicast Example: Router(config-vrf)# address-family ipv4 unicast	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 21	Do one of the following: <ul style="list-style-type: none"> redistribute connected [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] redistribute eigrp <i>process-id</i> [match { external internal }] [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] redistribute isis <i>process-id</i> [level { 1 1-inter-area 2 }] [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] redistribute ospf <i>process-id</i> [match { external [1 2] internal nssa-external [1 2] }] [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] redistribute ospfv3 <i>process-id</i> [match { external [1 2] internal nssa-external [1 2] }] [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] redistribute rip [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] 	(Optional) Configures redistribution of a protocol into the VRF address family context. Note Required if you are configuring multicast MVPN.

	Command or Action	Purpose
	<ul style="list-style-type: none"> • redistribute static [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] <p>Example:</p> <pre>Router(config-bgp-vrf-af)# redistribute eigrp 23</pre>	
Step 22	Use the commit or end command.	<p>commit —Saves the configuration changes and remains within the configuration session.</p> <p>end —Prompts user to take one of these actions:</p> <ul style="list-style-type: none"> • Yes — Saves configuration changes and exits the configuration session. • No —Exits the configuration session without committing the configuration changes. • Cancel —Remains in the configuration session, without committing the configuration changes.

Disable BGP Neighbor

Perform this task to administratively shut down a neighbor session without removing the configuration.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **shutdown**
5. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 127
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address*

Example:

```
Router(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **shutdown**

Example:

```
Router(config-bgp-nbr)# shutdown
```

Disables all active sessions for the specified neighbor.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Neighbor Capability Suppression

A BGP speaker can learn about BGP extensions that are supported by a peer by using the capabilities negotiation feature. Capabilities negotiation allows BGP to use only the set of features supported by both BGP peers on a link. The neighbor capability suppression feature will turn off neighbor capabilities negotiation during Open message exchange. This is required for interoperability with very old customer premises equipment devices that do not understand Capabilities option.

Configuration

Command introduced in neighbor, session-group and neighbor-group modes.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **capability suppress all**
5. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure Example: RP/0/RP0/CPU0:router# configure	Enters mode.
Step 2	router bgp <i>as-number</i> Example: Router(config)# router bgp 4	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	neighbor <i>ip-address</i> Example: Router(config-bgp)# neighbor 172.168.40.24	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	capability suppress all Example: Router(config-bgp-nbr)# capability suppress all	Turn off neighbor capabilities.
Step 5	Use the commit or end command.	commit —Saves the configuration changes and remains within the configuration session. end —Prompts user to take one of these actions: <ul style="list-style-type: none"> • Yes — Saves configuration changes and exits the configuration session. • No —Exits the configuration session without committing the configuration changes. • Cancel —Remains in the configuration session, without committing the configuration changes.

BGP Dynamic Neighbors

Earlier IOS-XR supported explicitly configured or static neighbor configuration. BGP dynamic neighbor support allows BGP peering to a group of remote neighbors that are defined by a range of IP addresses. Each range can be configured as a subnet IP address.

In larger BGP networks, implementing BGP dynamic neighbors can reduce the amount and complexity of CLI configuration and save CPU and memory usage. Both IPv4 and IPv6 peering are supported. Both IPv4 and IPv6 peering are supported.

Configuring BGP Dynamic Neighbors using Address Range

The existing neighbor command is extended to accept a prefix instead of an address.

In the following task, Router B is configured as a remote BGP peer. After a subnet range is configured, a TCP session is initiated by Router B which has an IP address in the subnet range and a new BGP neighbor is dynamically established.

After the initial configuration of subnet ranges and activation of the peer neighbor, dynamic BGP neighbor creation does not require any further CLI configuration on the Router A.



Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters the global configuration mode.

Step 2 **router bgp** *as-number*

Example:

```
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 3 **neighbor** *address prefix*

Example:

```
Router(config-bgp)# neighbor 10.0.0.0/16
```

Places the router in neighbor configuration mode for BGP routing and configures the BGP dynamic neighbor within the subnet range.

Note

All commands currently supported under a static neighbor, including address-family and inheritance using neighbor-group, session-group and af-group, will be supported for dynamic neighbor ranges with the exception of the following commands:

- session-open-mode
- local address

Step 4 **remote-as** *as-number*

Example:

```
Router(config-bgp-nbr)# remote-as 1
```

Creates a neighbor and assigns a remote autonomous system (AS) number to it.

Step 5 `update-source type interface-id`

Example:

```
Router(config-bgp-nbr)# update-source FourHundredGige 0/0/0/0
```

Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.

The type and interface-id arguments specify the type and ID number of the interface. Use the CLI help (?) to see a list of all the possible interface types and their ID numbers.

Step 6 `address-family ipv4 unicast`

Example:

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

Specifies the IPv4 unicast address family unicast and enters address family configuration mode.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

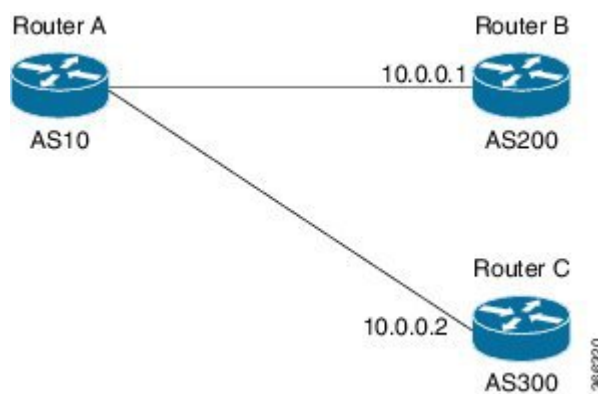
end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Remote AS List

In the following task, Router B and Router C are configured as a remote BGP peers. Both Router B and Router C are in different autonomous systems.

A list is created with the autonomous system of the remote routers and the list is then configured under neighbor mode using **remote-as-list** command.



Configuration

```
Router# configure
Router(config)# router bgp as-number
Router(config-bgp)# as-list name
Router(config-bgp)# neighbor address prefix
Router(config-bgp-nbr)# remote-as-list name
Router(config-bgp-nbr)# address-family ipv4 unicast
Router# commit
```

Maximum-peers and Idle-watch timeout

In the below task, maximum-peers and idle-watch timeout commands are configured for a remote BGP peer.

Procedure

Step 1 **configure****Example:**

```
Router# configure
```

Enters the global configuration mode.

Step 2 **router bgp** *as-number***Example:**

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

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *address prefix***Example:**

```
Router(config-bgp)# neighbor 10.0.0.0/16
```

Places the router in neighbor configuration mode for BGP routing and configures the BGP dynamic neighbor within the subnet range.

Step 4 **maximum-peers** *number***Example:**

```
Router(config-bgp-nbr)# maximum-peers 16
```

This is used to configure an upper limit on the number of dynamic neighbor instances allowed under a range.

Step 5 **idle-watch-time** *number***Example:**

```
Router(config-bgp)# idle-watch-time 120
```

Configures the time to wait before deleting an idle TCP instance.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Resetting Neighbors Using BGP Inbound Soft Reset

Perform this task to trigger an inbound soft reset of the specified address families for the specified group or neighbors. The group is specified by the ***, *ip-address*, *as-number*, or **external** keywords and arguments.

Resetting neighbors is useful if you change the inbound policy for the neighbors or any other configuration that affects the sending or receiving of routing updates. If an inbound soft reset is triggered, BGP sends a REFRESH request to the neighbor if the neighbor has advertised the ROUTE_REFRESH capability. To determine whether the neighbor has advertised the ROUTE_REFRESH capability, use the **show bgp neighbors** command.

Procedure

Step 1 **show bgp neighbors**

Example:

```
Router# show bgp neighbors
```

Verifies that received route refresh capability from the neighbor is enabled.

Step 2 **soft [in [prefix-filter] | out]**

Example:

```
Router# clear bgp ipv4 unicast 10.0.0.1 soft in
```

Soft resets a BGP neighbor.

- The *** keyword resets all BGP neighbors.
 - The *ip-address* argument specifies the address of the neighbor to be reset.
 - The *as-number* argument specifies that all neighbors that match the autonomous system number be reset.
 - The **external** keyword specifies that all external neighbors are reset.
-

Resetting Neighbors Using BGP Outbound Soft Reset

Perform this task to trigger an outbound soft reset of the specified address families for the specified group or neighbors. The group is specified by the *****, *ip-address*, *as-number*, or **external** keywords and arguments.

Resetting neighbors is useful if you change the outbound policy for the neighbors or any other configuration that affects the sending or receiving of routing updates.

If an outbound soft reset is triggered, BGP resends all routes for the address family to the given neighbors.

To determine whether the neighbor has advertised the ROUTE_REFRESH capability, use the **show bgp neighbors** command.

Procedure

Step 1 **show bgp neighbors**

Example:

```
Router# show bgp neighbors
```

Verifies that received route refresh capability from the neighbor is enabled.

Step 2 **clear bgp ipv4 unicast *ip-address* soft out**

Example:

```
Router# clear bgp ipv4 unicast 10.0.0.2 soft out
```

Soft resets a BGP neighbor.

- The ***** keyword resets all BGP neighbors.
 - The *ip-address* argument specifies the address of the neighbor to be reset.
 - The *as-number* argument specifies that all neighbors that match the autonomous system number be reset.
 - The **external** keyword specifies that all external neighbors are reset.
-

Reset Neighbors Using BGP Hard Reset

Perform this task to reset neighbors using a hard reset. 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.

Procedure

```
clear bgp { ipv4 { unicast | labeled-unicast | all | tunnel tunnel | mdt } | ipv6 unicast | all | labeled-unicast
} | all { unicast | multicast | all | labeled-unicast | mdt | tunnel } | vpnv4 unicast | vrf { vrf-name | all } {
ipv4 unicast | labeled-unicast } | ipv6 unicast } | vpnv6 unicast } { * | ip-address | as as-number | external
} [ graceful ] soft [ in [ prefix-filter ] | out ] clear bgp { ipv4 | ipv6 } { unicast | labeled-unicast }
```

Example:

```
Router# clear bgp ipv4 unicast 10.0.0.3 graceful soft out
```

Clears a BGP neighbor.

- The ***** keyword resets all BGP neighbors.
- The *ip-address* argument specifies the address of the neighbor to be reset.
- The *as-number* argument specifies that all neighbors that match the autonomous system number be reset.
- The **external** keyword specifies that all external neighbors are reset.

The **graceful** keyword specifies a graceful restart.

Clearing Caches, Tables, and Databases

Perform this task to remove all contents of a particular cache, table, or database. The **clear bgp** command resets the sessions of the specified group of neighbors (hard reset); it 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. Clearing a cache, table, or database can become necessary when the contents of the particular structure have become, or are suspected to be, invalid.

Procedure

Step 1 `clear bgp ipv4 ip-address`

Example:

```
Router# clear bgp ipv4 172.20.1.1
```

Clears a specified neighbor.

Step 2 `clear bgp external`

Example:

```
Router# clear bgp external
```

Clears all external peers.

Step 3 **clear bgp *****Example:**

```
Router# clear bgp *
```

Clears all BGP neighbors.

Display System and Network Statistics

Perform this task to display specific statistics, such as the contents of BGP routing tables, caches, and databases. Information provided can be used to determine resource usage and solve network problems. You can also display information about node reachability and discover the routing path that the packets of your device are taking through the network.

SUMMARY STEPS

1. **show bgp cidr-only**
2. **show bgp community** *community-list* [**exact-match**]
3. **show bgp regexp** *regular-expression*
4. **show bgp**
5. **show bgp neighbors** *ip-address* [**advertised-routes** | **dampened-routes** | **flap-statistics** | **performance-statistics** | **received** *prefix-filter* | **routes**]
6. **show bgp paths**
7. **show bgp neighbor-group** *group-name* **configuration**
8. **show bgp summary**

DETAILED STEPS

Procedure

Step 1 **show bgp cidr-only****Example:**

```
Router# show bgp cidr-only
```

Displays routes with nonnatural network masks (classless interdomain routing [CIDR]) routes.

Step 2 **show bgp community** *community-list* [**exact-match**]**Example:**

```
Router# show bgp community 1081:5 exact-match
```

Displays routes that match the specified BGP community.

Step 3 **show bgp regexp** *regular-expression*

Example:

```
Router# show bgp regexp "^3 "
```

Displays routes that match the specified autonomous system path regular expression.

Step 4 **show bgp****Example:**

```
Router# show bgp
```

Displays entries in the BGP routing table.

Step 5 **show bgp neighbors** *ip-address* [**advertised-routes** | **dampened-routes** | **flap-statistics** | **performance-statistics** | **received** *prefix-filter* | **routes**]**Example:**

```
Router# show bgp neighbors 10.0.101.1
```

Displays information about the BGP connection to the specified neighbor.

- The **advertised-routes** keyword displays all routes the router advertised to the neighbor.
- The **dampened-routes** keyword displays the dampened routes that are learned from the neighbor.
- The **flap-statistics** keyword displays flap statistics of the routes learned from the neighbor.
- The **performance-statistics** keyword displays performance statistics relating to work done by the BGP process for this neighbor.
- The **received** *prefix-filter* keyword and argument display the received prefix list filter.
- The **routes** keyword displays routes learned from the neighbor.

Step 6 **show bgp paths****Example:**

```
Router# show bgp paths
```

Displays all BGP paths in the database.

Step 7 **show bgp neighbor-group** *group-name* **configuration****Example:**

```
Router# show bgp neighbor-group group_1 configuration
```

Displays the effective configuration for a specified neighbor group, including any configuration inherited by this neighbor group.

Step 8 **show bgp summary****Example:**

```
Router# show bgp summary
```

Displays the status of all BGP connections.

Display BGP Process Information

Perform this task to display specific BGP process information.

Procedure

Step 1 **show bgp process****Example:**

```
Router# show bgp process
```

Displays status and summary information for the BGP process. The output shows various global and address family-specific BGP configurations. A summary of the number of neighbors, update messages, and notification messages sent and received by the process is also displayed.

Step 2 **show bgp ipv4 unicast summary****Example:**

```
Router# show bgp ipv4 unicast summary
```

Displays a summary of the neighbors for the IPv4 unicast address family.

Step 3 **show bgp vpnv4 unicast summary****Example:**

```
Router# show bgp vpnv4 unicast summary
```

Displays a summary of the neighbors for the VPNv4 unicast address family.

Step 4 **show bgp vrf (*vrf-name* | all)****Example:**

```
Router# show bgp vrf vrf_A
```

Displays BGP VPN virtual routing and forwarding (VRF) information.

Step 5 **show bgp process detail****Example:**

```
Router# show bgp processes detail
```

Displays detailed process information including the memory used by each of various internal structure types.

Step 6 **show bgp summary****Example:**

```
Router# show bgp summary
```

Displays the status of all BGP connections.

Step 7 **show placement program bgp****Example:**

```
Router# show placement program bgp
```

Displays BGP program information.

- If a program is shown as having ‘rejected locations’ (for example, locations where program cannot be placed), the locations in question can be viewed using the **show placement program bgp** command.
- If a program has been placed but not started, the amount of elapsed time since the program was placed is displayed in the Waiting to start column.

Step 8 **show placement program brib**

Example:

```
Router# show placement program brib
```

Displays bRIB program information.

- If a program is shown as having ‘rejected locations’ (for example, locations where program cannot be placed), the locations in question can be viewed using the **show placement program bgp** command.
- If a program has been placed but not started, the amount of elapsed time since the program was placed is displayed in the Waiting to start column.

Configure iBGP Multipath Load Sharing

Perform this task to configure the iBGP Multipath Load Sharing:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** {**ipv4**|**ipv6**} {**unicast**|**multicast**}
4. **maximum-paths ibgp** *number*
5. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:

```
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 3 **address-family {ipv4|ipv6} {unicast|multicast}**

Example:

```
Router(config-bgp)# address-family ipv4 multicast
```

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

Step 4 **maximum-paths ibgp *number***

Example:

```
Router(config-bgp-af)# maximum-paths ibgp 30
```

Configures the maximum number of iBGP paths for load sharing.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

iBGP Multipath Loadsharing Configuration: Example

The following is a sample configuration where 30 paths are used for loadsharing:

```
router bgp 100
  address-family ipv4 multicast
    maximum-paths ibgp 30
  !
!
end
```

Originate Prefixes with AiGP

Perform this task to configure origination of routes with the AiGP metric:

Before you begin

Origination of routes with the accumulated interior gateway protocol (AiGP) metric is controlled by configuration. AiGP attributes are attached to redistributed routes that satisfy following conditions:

- The protocol redistributing the route is enabled for AiGP.

- The route is an interior gateway protocol (iGP) route redistributed into border gateway protocol (BGP). The value assigned to the AiGP attribute is the value of iGP next hop to the route or as set by a route-policy.
- The route is a static route redistributed into BGP. The value assigned is the value of next hop to the route or as set by a route-policy.
- The route is imported into BGP through network statement. The value assigned is the value of next hop to the route or as set by a route-policy.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **route-policy *aigp_policy***

Example:

```
Router(config)# route-policy aip_policy
```

Enters route-policy configuration mode and sets the route-policy

Step 3 **set aigp-metric *igp-cost***

Example:

```
Router(config-rpl)# set aigp-metric igp-cost
```

Sets the internal routing protocol cost as the aigp metric.

Step 4 **exit**

Example:

```
Router(config-rpl)# exit
```

Exits route-policy configuration mode.

Step 5 **router bgp *as-number***

Example:

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

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 6 **address-family {*ipv4* | *ipv6*} *unicast***

Example:

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

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

Step 7 **redistribute ospf *osp* route-policy *plcy_name* metric *value***

Example:

```
Router(config-bgp-af)#redistribute ospf osp route-policy aigp_policy metric 1
```

Allows the redistribution of AiBGP metric into OSPF.

Step 8 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Originating Prefixes With AiGP: Example

The following is a sample configuration for originating prefixes with the AiGP metric attribute:

```
route-policy aigp-policy
  set aigp-metric 4
  set aigp-metric igp-cost
end-policy
!
router bgp 100
address-family ipv4 unicast
  network 10.2.3.4/24 route-policy aigp-policy
  redistribute ospf osp1 metric 4 route-policy aigp-policy
!
end
```

Configure BGP Accept Own

Perform this task to configure BGP Accept Own:

Procedure

Step 1 **configure**

Step 2 **router bgp** *as-number*

Example:

```
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 3 **neighbor** *ip-address*

Example:

```
Router(config-bgp)#neighbor 10.1.2.3
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **remote-as** *as-number***Example:**

```
Router(config-bgp-nbr)#remote-as 100
```

Assigns a remote autonomous system number to the neighbor.

Step 5 **update-source** *type interface-path-id***Example:**

```
Router(config-bgp-nbr)#update-source Loopback0
```

Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.

Step 6 **address-family** {*vpn4 unicast* | *vpn6 unicast*}**Example:**

```
Router(config-bgp-nbr)#address-family vpnv6 unicast
```

Specifies the address family as VPNv4 or VPNv6 and enters neighbor address family configuration mode.

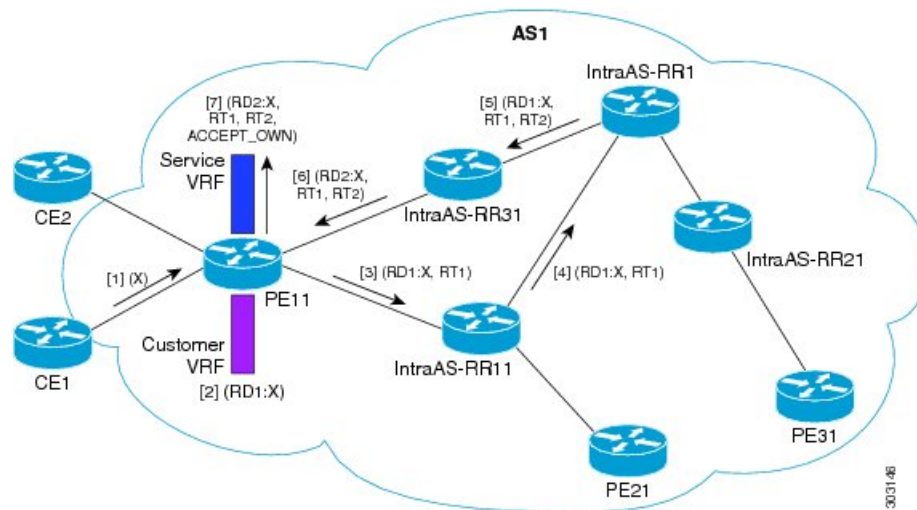
Step 7 **accept-own** [**inheritance-disable**]**Example:**

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

Enables handling of self-originated VPN routes containing Accept_Own community.

Use the **inheritance-disable** keyword to disable the "accept own" configuration and to prevent inheritance of "acceptown" from a parent configuration.

BGP Accept Own Configuration: Example



In this configuration example:

- PE11 is configured with Customer VRF and Service VRF.
- OSPF is used as the IGP.
- VPNv4 unicast and VPNv6 unicast address families are enabled between the PE and RR neighbors and IPv4 and IPv6 are enabled between PE and CE neighbors.

The Accept Own configuration works as follows:

1. CE1 originates prefix X.
2. Prefix X is installed in customer VRF as (RD1:X).
3. Prefix X is advertised to IntraAS-RR11 as (RD1:X, RT1).
4. IntraAS-RR11 advertises X to InterAS-RR1 as (RD1:X, RT1).
5. InterAS-RR1 attaches RT2 to prefix X on the inbound and ACCEPT_OWN community on the outbound and advertises prefix X to IntraAS-RR31.
6. IntraAS-RR31 advertises X to PE11.
7. PE11 installs X in Service VRF as (RD2:X, RT1, RT2, ACCEPT_OWN).

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

```
router bgp 100
 neighbor 45.1.1.1
  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 45.1.1.1
  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
!

```

Configuring BGP Link-state

To exchange BGP link-state (LS) information with a BGP neighbor, perform these steps:

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number*

Example:


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

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address*

Example:

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

Configures a CE neighbor. The ip-address argument must be a private address.

Step 4 **remote-as** *as-number*

Example:

```
Router(config-bgp-nbr)# remote-as 1
```

Configures the remote AS for the CE neighbor.

Step 5 **address-family link-state link-state**

Example:

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

Distributes BGP link-state information to the specified neighbor.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP Permanent Network

Perform this task to configure BGP permanent network. You must configure at least one route-policy to identify the set of prefixes (networks) for which the permanent network (path) is to be configured.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **prefix-set** *prefix-set-name***Example:**

```
Router(config)# prefix-set PERMANENT-NETWORK-IPv4
Router(config-pfx)# 1.1.1.1/32,
Router(config-pfx)# 2.2.2.2/32,
Router(config-pfx)# 3.3.3.3/32
Router(config-pfx)# end-set
```

Enters prefix set configuration mode and defines a prefix set for contiguous and non-contiguous set of bits.

Step 3 **exit****Example:**

```
Router(config-pfx)# exit
```

Exits prefix set configuration mode and enters global configuration mode.

Step 4 **route-policy** *route-policy-name***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
```

Creates a route policy and enters route policy configuration mode, where you can define the route policy.

Step 5 **end-policy****Example:**

```
Router(config-rpl)# end-policy
```

Ends the definition of a route policy and exits route policy configuration mode.

Step 6 **router bgp** *as-number***Example:**

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

Specifies the autonomous system number and enters the BGP configuration mode.

Step 7 **address-family { ipv4 | ipv6 } unicast****Example:**

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

Step 8 **permanent-network route-policy route-policy-name**

Example:

```
Router(config-bgp-af)# permanent-network route-policy POLICY-PERMANENT-NETWORK-IPv4
```

Configures the permanent network (path) for the set of prefixes as defined in the route-policy.

Step 9 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 10 **show bgp {ipv4 | ipv6} unicast prefix-set**

Example:

```
show bgp ipv4 unicast
```

(Optional) Displays whether the prefix-set is a permanent network in BGP.

How to Advertise Permanent Network

Perform this task to identify the peers to whom the permanent paths must be advertised.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp as-number**

Example:

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

Specifies the autonomous system number and enters the BGP configuration mode.

Step 3 **neighbor** *ip-address*

Example:

```
Router(config-bgp)# neighbor 10.255.255.254
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **remote-as** *as-number*

Example:

```
Router(config-bgp-nbr)# remote-as 4713
```

Assigns the neighbor a remote autonomous system number.

Step 5 **address-family { ipv4 | ipv6 } unicast**

Example:

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

Step 6 **advertise permanent-network**

Example:

```
Router(config-bgp-nbr-af)# advertise permanent-network
```

Specifies the peers to whom the permanent network (path) is advertised.

Step 7 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 8 **show bgp {ipv4 | ipv6} unicast neighbor** *ip-address*

Example:

```
Routershow bgp ipv4 unicast neighbor 10.255.255.254
```

(Optional) Displays whether the neighbor is capable of receiving BGP permanent networks.

Enable BGP Unequal Cost Recursive Load Balancing

Procedure

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: Router(config)# router bgp 120	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	address-family { ipv4 ipv6 } unicast Example: Router(config-bgp)# address-family ipv4 unicast	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submenu. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	maximum-paths { ebgp ibgp eibgp } maximum [unequal-cost] Example: Router(config-bgp-af)# maximum-paths ebgp 3	Configures the maximum number of parallel routes that BGP installs in the routing table. <ul style="list-style-type: none"> • ebgp maximum : Consider only eBGP paths for multipath. • ibgp maximum [unequal-cost] : Consider load balancing between iBGP learned paths. • eibgp maximum : Consider both eBGP and iBGP learned paths for load balancing. eiBGP load balancing always does unequal-cost load balancing. When eiBGP is applied, eBGP or iBGP load balancing cannot be configured; however, eBGP and iBGP load balancing can coexist.
Step 5	exit Example: Router(config-bgp-af)# exit	Exits the current configuration mode.
Step 6	neighbor ip-address Example: Router(config-bgp)# neighbor 10.0.0.0	Configures a CE neighbor. The <i>ip-address</i> argument must be a private address.
Step 7	dmz-link-bandwidth Example: Router(config-bgp-nbr)# dmz-link-bandwidth	Originates a demilitarized-zone (DMZ) link-bandwidth extended community for the link to an eBGP/iBGP neighbor.

	Command or Action	Purpose
Step 8	commit	

BGP Unequal Cost Recursive Load Balancing: Example

This is a sample configuration for unequal cost recursive load balancing:

```

interface Loopback0
  ipv4 address 20.20.20.20 255.255.255.255
!
interface MgmtEth0/RSP0/CPU0/0
  ipv4 address 8.43.0.10 255.255.255.0
!
interface TenGigE0/3/0/0
  bandwidth 8000000
  ipv4 address 11.11.11.11 255.255.255.0
  ipv6 address 11:11:0:1::11/64
!
interface TenGigE0/3/0/1
  bandwidth 7000000
  ipv4 address 11.11.12.11 255.255.255.0
  ipv6 address 11:11:0:2::11/64
!
interface TenGigE0/3/0/2
  bandwidth 6000000
  ipv4 address 11.11.13.11 255.255.255.0
  ipv6 address 11:11:0:3::11/64
!
interface TenGigE0/3/0/3
  bandwidth 5000000
  ipv4 address 11.11.14.11 255.255.255.0
  ipv6 address 11:11:0:4::11/64
!
interface TenGigE0/3/0/4
  bandwidth 4000000
  ipv4 address 11.11.15.11 255.255.255.0
  ipv6 address 11:11:0:5::11/64
!
interface TenGigE0/3/0/5
  bandwidth 3000000
  ipv4 address 11.11.16.11 255.255.255.0
  ipv6 address 11:11:0:6::11/64
!
interface TenGigE0/3/0/6
  bandwidth 2000000
  ipv4 address 11.11.17.11 255.255.255.0
  ipv6 address 11:11:0:7::11/64
!
interface TenGigE0/3/0/7
  bandwidth 1000000
  ipv4 address 11.11.18.11 255.255.255.0
  ipv6 address 11:11:0:8::11/64
!
interface TenGigE0/4/0/0
  description CONNECTED TO IXIA 1/3
  transceiver permit pid all
!
interface TenGigE0/4/0/2
  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 20.20.20.20
  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 11.11.13.12
    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
!
!
end

```

Configuring BGP Large Communities

BGP communities provide a way to group destinations and apply routing decisions such as acceptance, rejection, preference, or redistribution on a group of destinations using community attributes. BGP community attributes are variable length attributes consisting of a set of one or more 4-byte values which are split into two parts of 16 bits. The higher-order 16 bits represents the AS number and the lower order bits represents a locally defined value assigned by the operator of the AS.

Since the adoption of 4-byte ASNs (RFC6793), the BGP communities attribute can no longer accommodate the 4 byte ASNs as you need more than 4 bytes to encode the 4-byte ASN and an AS specific value that you want to tag with the route. Although BGP extended community permits a 4-byte AS to be encoded as the global administrator field, the local administrator field has only 2-byte of available space. So, 6-byte extended community attribute is also unsuitable. To overcome this limitation, you can configure a 12-byte BGP large community which is an optional attribute that provides the most significant 4-byte value to encode autonomous system number as the global administrator and the remaining two 4-byte assigned numbers to encode the local values.

Similar to BGP communities, routers can apply BGP large communities to BGP routes by using route policy languages (RPL) and other routers can then perform actions based on the community that is attached to the route. The policy language provides sets as a container for groups of values for matching purposes.

When large communities are specified in other commands, they are specified as three non negative decimal integers separated by colons. For example, 1:2:3. Each integer is stored in 32 bits. The possible range for each integer is 0 to 4294967295.

In route-policy statements, each integer in the BGP large community can be replaced by any of the following expressions :

- [x..y] — This expression specifies a range between x and y, inclusive.
- * —This expression stands for any number.
- peeras — This expression is replaced by the AS number of the neighbor from which the community is received or to which the community is sent, as appropriate.
- not-peeras —This expression matches any number other than the peeras.

- **private-as** — This expression specifies any number in the private ASN range: [64512..65534] and [4200000000..4294967294].

These expressions can be also used in policy-match statements.

IOS regular expression (**ios-regex**) and DFA style regular expression (**dfa-regex**) can be used in any of the large-community policy match and delete statements. For example, the IOS regular expression **ios-regex '^5.:7\$'** is equivalent to the expression **5.:7**.

The **send-community-ebgp** command is extended to include BGP large communities. This command is required for the BGP speaker to send large communities to ebgp neighbors.

Restrictions and Guidelines

The following restrictions and guidelines apply for BGP large communities:

- All functionalities of 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.
- The **peer-as** expression cannot be used in a large-community-set.
- The **peer-as** expression can only be used in large-community match or delete statements that appear in route policies that are applied at the neighbor-in or neighbor-out attach points.
- The **not-peer-as** expression cannot be used in a large-community-set or in policy set statements.

Configuration Example: Large Community Set

A large-community set defines a set of large communities. Named large-community sets are used in route-policy match and set statements.

This example shows how to create a named large-community set.

```
RP/0/RP0/CPU0:router(config)# large-community-set catbert
RP/0/RP0/CPU0:router(config-largecomm)# 1: 2: 3,
RP/0/RP0/CPU0:router(config-largecomm)# peer-as:2:3
RP/0/RP0/CPU0:router(config-largecomm)# end-set
```

Configuration Example: Set Large Community

The following example shows how to set the BGP large community attribute in a route, using the **set large-community {large-community-set-name | inline-large-community-set | parameter} [additive]** command. You can specify a named large-community-set or an inline set. The **additive** keyword retains the large communities already present in the route and adds the new set of large communities. However the **additive** keyword does not result in duplicate entries.

If a particular large community is attached to a route and you specify the same large community again with the additive keyword in the set statement, then the specified large community is not added again. The merging operation removes duplicate entries. This also applies to the **peer-as** keyword.

The **peer-as** expression in the example is replaced by the AS number of the neighbor from which the BGP large community is received or to which the community is sent, as appropriate.

```
RP/0/RP0/CPU0:router(config)# route-policy mordac
RP/0/RP0/CPU0:router(config-rpl)# set large-community (1:2:3, peer-as:2:3)
```

```

RP/0/RP0/CPU0:router(config-rpl)# end-set
RP/0/RP0/CPU0:router(config)# large-community-set catbert
RP/0/RP0/CPU0:router(config-largecomm)# 1: 2: 3,
RP/0/RP0/CPU0:router(config-largecomm)# peers:2:3
RP/0/RP0/CPU0:router(config-largecomm)# end-set
RP/0/RP0/CPU0:router(config)# route-policy wally
RP/0/RP0/CPU0:router(config-rpl)# set large-community catbert additive
RP/0/RP0/CPU0:router(config-rpl)# end-set

```

In this example, if the route-policy mordac is applied to a neighbor, the ASN of which is 1, then the large community (1:2:3) is set only once.



Note You should configure the **send-community-ebgp** command to send large communities to ebgp neighbors.

Configuration Example: Large Community Matches-any

The following example shows how to configure a route policy to match any element of a large -community set. This is a boolean condition and returns true if any of the large communities in the route match any of the large communities in the match condition.

```

RP/0/RP0/CPU0:router(config)# route-policy elbonia
RP/0/RP0/CPU0:router(config-rpl)# if large-community matches-any (1:2:3, 4:5:*) then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 94
RP/0/RP0/CPU0:router(config-rpl)# endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy

```

Configuration Example: Large Community Matches-every

The following example shows how to configure a route policy where every match specification in the statement must be matched by at least one large community in the route.

```

RP/0/RP0/CPU0:router(config)# route-policy bob
RP/0/RP0/CPU0:router(config-rpl)# if large-community matches-every (*:3, 4:5:*) then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 94
RP/0/RP0/CPU0:router(config-rpl)# endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy

```

In this example, routes with these sets of large communities return TRUE:

- (1:1:3, 4:5:10)
- (4:5:3) —This single large community matches both specifications.
- (1:1:3, 4:5:10, 7:6:5)

Routes with the following set of large communities return FALSE:

(1:1:3, 5:5:10)—The specification (4:5:*) is not matched.

Configuration Example: Large Community Matches-within

The following example shows how to configure a route policy to match within a large community set. This is similar to the **large-community matches-any** command but every large community in the route must match at least one match specification. Note that if the route has no large communities, then it matches.

```

RP/0/RP0/CPU0:router(config)# route-policy bob
RP/0/RP0/CPU0:router(config-rpl)# if large-community matches-within (*:3, 4:5:*) then

```

```
RP/0/RP0/CPU0:router(config-rpl)# set local-preference 103
RP/0/RP0/CPU0:router(config-rpl)# endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

For example, routes with these sets of large communities return TRUE:

- (1:1:3, 4:5:10)
- (4:5:3)
- (1:2:3, 6:6:3, 9:4:3)

Routes with this set of large communities return FALSE:

(1:1:3, 4:5:10, 7:6:5) —The large community (7:6:5) does not match

Configuration Example: Community Matches-within

The following example shows how to configure a route policy to match within the elements of a community set. This command is similar to the **community matches-any** command, but every community in the route must match at least one match specification. If the route has no communities, then it matches.

```
RP/0/RP0/CPU0:router(config)# route-policy bob
RP/0/RP0/CPU0:router(config-rpl)# if community matches-within (*:3, 5:*) then
RP/0/RP0/CPU0:router(config-rpl)# set local-preference 94
RP/0/RP0/CPU0:router(config-rpl)# endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

For example, routes with these sets of communities return TRUE:

- (1:3, 5:10)
- (5:3)
- (2:3, 6:3, 4:3)

Routes with this set of communities return FALSE:

(1:3, 5:10, 6:5) —The community (6:5) does not match.

Configuration Example: Large Community Is-empty

The following example shows using the **large-community is-empty** clause to filter routes that do not have the large-community attribute set.

```
RP/0/RP0/CPU0:router(config)# route-policy lrg_comm_rp4
RP/0/RP0/CPU0:router(config-rpl)# if large-community is-empty then
RP/0/RP0/CPU0:router(config-rpl)# set local-preference 104
RP/0/RP0/CPU0:router(config-rpl)# endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

Configuration Example: Attribute Filter Group

The following example shows how to configure and apply the attribute-filter group with large-community attributes for a BGP neighbor. The filter specifies the BGP path attributes and an action to take when BGP update message is received. If an update message is received from the BGP neighbor that contains any of the specified attributes, then the specified action is taken. In this example, the attribute filter named dogbert is created and applied to the BGP neighbor 10.0.1.101. It specifies the large community attribute and the action

of discard. That means, if the large community BGP path attribute is received in a BGP UPDATE message from the neighbor 10.0.1.101 then the attribute will be discarded before further processing of the message.

```
RP/0/RP0/CPU0:router(config)# router bgp 100
RP/0/RP0/CPU0:router(config-bgp)# attribute-filter group dogbert
RP/0/RP0/CPU0:router(config-bgp-attrfg)# attribute LARGE-COMMUNITY discard
RP/0/RP0/CPU0:router(config-bgp-attrfg)# neighbor 10.0.1.101
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 6461
RP/0/RP0/CPU0:router(config-bgp-nbr)# update in filtering
RP/0/RP0/CPU0:router(config-nbr-upd-filter)# attribute-filter group dogbert
```

Configuration Example: Deleting Large Community

The following example shows how to delete specified BGP large-communities from a route policy using the **delete large-community** command.

```
RP/0/RP0/CPU0:router(config)# route-policy lrg_comm_rp2
RP/0/RP0/CPU0:router(config-rpl)# delete large-community in (ios-regex '^100000:')
RP/0/RP0/CPU0:router(config-rpl)# delete large-community all
RP/0/RP0/CPU0:router(config-rpl)# delete large-community not in (peeras:*:, 41289:*:*)
```

Verification

This example displays the routes with large-communities given in the **show bgp large-community list-of-large-communities [exact-match]** command. If the optional keyword **exact-match** is used, then the listed routes will contain only the specified large communities. Otherwise, the displayed routes may contain additional large communities.

```
RP/0/0/CPU0:R1# show bgp large-community 1:2:3 5:6:7
Thu Mar 23 14:40:33.597 PDT
BGP router identifier 4.4.4.4, 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 ?
```

This example displays the large community attached to a network using the **show bgp ip-address/prefix-length** command.

```
RP/0/0/CPU0:R4# 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

```

Enabling BGP: Example

The following shows how to enable BGP.

```

prefix-set static
  2020::/64,
  2012::/64,
  10.10.0.0/16,
  10.2.0.0/24
end-set

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 2003::121
end-policy

route-policy set_next_hop_static_v6
  if (destination in static) then
    set next-hop 2011::121
  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 1.1.1.1
  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 2012::/64 route-policy set_next_hop_agg_v6
  aggregate-address 2013::/64
  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

```

Displaying BGP Update Groups: Example

The following is sample output from the **show bgp update-group**:

show bgp update-group

```

Update group for IPv4 Unicast, index 0.1:
  Attributes:
    Outbound Route map:rm
    Minimum advertisement interval:30
  Messages formatted:2, replicated:2
  Neighbors in this update group:
    10.0.101.92

Update group for IPv4 Unicast, index 0.2:
  Attributes:
    Minimum advertisement interval:30
  Messages formatted:2, replicated:2
  Neighbors in this update group:
    10.0.101.91

```

BGP Neighbor Configuration: Example

The following example shows how BGP neighbors on an autonomous system are configured to share information. In the example, a BGP router is assigned to autonomous system 109, and two networks are listed as originating in the autonomous system. Then the addresses of three remote routers (and their autonomous systems) are listed. The router being configured shares information about networks 172.16.0.0 and 192.168

.7.0 with the neighbor routers. The first router listed is in a different autonomous system; the second **neighbor** and **remote-as** commands specify an internal neighbor (with the same autonomous system number) at address 172.26.234.2; and the third **neighbor** and **remote-as** commands specify a neighbor on a different autonomous system.

```
route-policy pass-all
  pass
end-policy
router bgp 109
  address-family ipv4 unicast
    network 172.16.0.0 255.255.0.0
    network 192.168.7.0 255.255.0.0
    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
```

BGP Confederation: Example

The following is a sample configuration that shows several peers in a confederation. The confederation consists of three internal autonomous systems with autonomous system numbers 6001, 6002, and 6003. To the BGP speakers outside the confederation, the confederation looks like a normal autonomous system with autonomous system number 666 (specified using the **bgp confederation identifier** command).

In a BGP speaker in autonomous system 6001, the **bgp confederation peers** command marks the peers from autonomous systems 6002 and 6003 as special eBGP peers. Hence, peers 171.16.232.55 and 171.16.232.56 get the local preference, next hop, and MED unmodified in the updates. The router at 171.19.69.1 is a normal eBGP speaker, and the updates received by it from this peer are just like a normal eBGP update from a peer in autonomous system 666.

```
router bgp 6001
  bgp confederation identifier 666
  bgp confederation peers
    6002
    6003
  exit
  address-family ipv4 unicast
    neighbor 171.16.232.55
      remote-as 6002
    exit
  address-family ipv4 unicast
    neighbor 171.16.232.56
      remote-as 6003
    exit
  address-family ipv4 unicast
```

```
neighbor 171.19.69.1
remote-as 777
```

In a BGP speaker in autonomous system 6002, the peers from autonomous systems 6001 and 6003 are configured as special eBGP peers. Peer 171.17.70.1 is a normal iBGP peer, and peer 199.99.99.2 is a normal eBGP peer from autonomous system 700.

```
router bgp 6002
  bgp confederation identifier 666
  bgp confederation peers
    6001
    6003
  exit
  address-family ipv4 unicast
    neighbor 171.17.70.1
    remote-as 6002
  exit
  address-family ipv4 unicast
    neighbor 171.19.232.57
    remote-as 6001
  exit
  address-family ipv4 unicast
    neighbor 171.19.232.56
    remote-as 6003
  exit
  address-family ipv4 unicast
    neighbor 171.19.99.2
    remote-as 700
  exit
  address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-all out
```

In a BGP speaker in autonomous system 6003, the peers from autonomous systems 6001 and 6002 are configured as special eBGP peers. Peer 192.168.200.200 is a normal eBGP peer from autonomous system 701.

```
router bgp 6003
  bgp confederation identifier 666
  bgp confederation peers
    6001
    6002
  exit
  address-family ipv4 unicast
    neighbor 171.19.232.57
    remote-as 6001
  exit
  address-family ipv4 unicast
    neighbor 171.19.232.55
    remote-as 6002
  exit
  address-family ipv4 unicast
    neighbor 192.168.200.200
    remote-as 701
  exit
  address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-all out
```


The following is a part of the configuration from the BGP speaker 192.168.200.205 from autonomous system 701 in the same example. Neighbor 172.16.232.56 is configured as a normal eBGP speaker from autonomous system 666. The internal division of the autonomous system into multiple autonomous systems is not known to the peers external to the confederation.

```
router bgp 701
 address-family ipv4 unicast
  neighbor 172.16.232.56
  remote-as 666
 exit
 address-family ipv4 unicast
 route-policy pass-all in
 route-policy pass-all out
 exit
 address-family ipv4 unicast
 neighbor 192.168.200.205
 remote-as 701
```

BGP Route Reflector: Example

The following example shows how to use an address family to configure internal BGP peer 10.1.1.1 as a route reflector client:

```
router bgp 140
 address-family ipv4 unicast
  neighbor 10.1.1.1
  remote-as 140
 address-family ipv4 unicast
  route-reflector-client
 exit
```

BGP Route Reflector: Example

The following example shows how to use an address family to configure internal BGP peer 10.1.1.1 as a route reflector client:

```
router bgp 140
 address-family ipv4 unicast
  neighbor 10.1.1.1
  remote-as 140
 address-family ipv4 unicast
  route-reflector-client
 exit
```

BGP MDT Address Family Configuration: Example

The following example shows how to configure an MDT address family in BGP:

```
router bgp 10

  bgp router-id 10.0.0.2
  address-family ipv4 unicast
  address-family vpnv4 unicast
  address-family ipv4 mdt

  !
  neighbor 1.1.1.1

  remote-as 11
  update-source Loopback0
  address-family ipv4 unicast
  address-family vpnv4 unicast
  address-family ipv4 md

  !
```

BGP Nonstop Routing Configuration: Example

The following example shows how to enable BGP NSR:

```
configure
router bgp 120
nsr
end
```

The following example shows how to disable BGP NSR:

```
configure
router bgp 120
no nsr
end
```

Best-External Path Advertisement Configuration: Example

The following example shows how to configure Best-External Path Advertisement:

```
router bgp 100
  address-family l2vpn vpls-vpws
  advertise best-external
end
```

Primary Backup Path Installation: Example

The following example shows how to enable installation of primary backup path:

```
router bgp 100
 address-family l2vpn vpls-vpws
  additional-paths install backup
end
```

iBGP Multipath Loadsharing Configuration: Example

The following is a sample configuration where 30 paths are used for loadsharing:

```
router bgp 100
 address-family ipv4 multicast
  maximum-paths ibgp 30
!
!
end
```

Discard Extra Paths Configuration: Example

The following example shows how to configure discard extra paths feature for the IPv4 address family:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# router bgp 10
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# maximum-prefix 1000 discard-extra-paths
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# commit
```

Verify Per Neighbor TCP MSS: Examples

The following example shows how to verify the per neighbor TCP MSS feature on a router:

The **show bgp neighbor** output shows the cumulative number for the *Prefix advertised* count if the same prefixes are withdrawn and re-advertised.

```
Router#show bgp neighbor 10.0.0.2

BGP neighbor is 10.0.0.2
Remote AS 1, local AS 1, internal link
Remote router ID 10.0.0.2
BGP state = Established, up for 00:09:17
Last read 00:00:16, 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 00:00:16, attempted 19, written 19
```

```

Second last write 00:01:16, 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 Dec 7 11:58:42.411 last full not set pulse count 23
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
Multi-protocol capability received
Neighbor capabilities:
Route refresh: advertised (old + new) and received (old + new)
Graceful Restart (GR Awareness): advertised and received
4-byte AS: advertised and received
Address family IPv4 Unicast: advertised and received
Received 12 messages, 0 notifications, 0 in queue
Sent 12 messages, 0 notifications, 0 in queue
Minimum time between advertisement runs is 0 secs
TCP Maximum Segment Size 500

For Address Family: IPv4 Unicast
BGP neighbor version 4
Update group: 0.2 Filter-group: 0.1 No Refresh request being processed
Route refresh request: received 0, sent 0
0 accepted prefixes, 0 are bestpaths
Cumulative no. of prefixes denied: 0.
Prefix advertised 0, suppressed 0, withdrawn 0
Maximum prefixes allowed 1048576
Threshold for warning message 75%, restart interval 0 min
AIGP is enabled
An EoR was received during read-only mode
Last ack version 4, Last synced ack version 0
Outstanding version objects: current 0, max 0
Additional-paths operation: None
Send Multicast Attributes

```

The following example shows how to verify the TCP MSS configuration:

```

RP/0/0/CPU0:ios#show bgp neighbor 10.0.0.2 configuration

neighbor 10.0.0.2
remote-as 1 []
tcp-mss 400 [n:n1]
address-family IPv4 Unicast []

```

The following example shows how to display TCP connection endpoints information:

```

RP/0/0/CPU0:ios#show tcp brief

```

PCB	VRF-ID	Recv-Q	Send-Q	Local Address	Foreign Address	State
0x08789b28	0x60000000	0	0	:::179	:::0	LISTEN
0x08786160	0x00000000	0	0	:::179	:::0	LISTEN
0xecb0c9f8	0x60000000	0	0	10.0.0.1:12404	10.0.0.2:179	ESTAB
0x0878b168	0x60000000	0	0	11.0.0.1:179	11.0.0.2:61177	ESTAB
0xecb0c6b8	0x60000000	0	0	0.0.0.0:179	0.0.0.0:0	LISTEN
0x08781590	0x00000000	0	0	0.0.0.0:179	0.0.0.0:0	LISTEN

The following example shows how to display TCP connection information for a specific PCB value:

```
RP/0/0/CPU0:ios#show tcp pcb 0xecb0c9f8
```

```
Connection state is ESTAB, I/O status: 0, socket status: 0
Established at Sun Dec 7 11:49:39 2014
```

```
PCB 0xecb0c9f8, SO 0xecb01b68, TCPCB 0xecb01d78, vrfid 0x60000000,
Pak Prio: Medium, TOS: 192, TTL: 255, Hash index: 1322
Local host: 10.0.0.1, Local port: 12404 (Local App PID: 19840)
Foreign host: 10.0.0.2, Foreign port: 179
```

```
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 17 2 0
SendWnd 0 0 0
TimeWait 0 0 0
AckHold 13 5 0
KeepAlive 1 0 0
PmtuAger 0 0 0
GiveUp 0 0 0
Throttle 0 0 0
```

```
iss: 1728179225 snduna: 1728179536 sndnxt: 1728179536
sndmax: 1728179536 sndwnd: 32517 sndcwnd: 1000
irs: 2055835995 rcvnxt: 2055836306 rcvwnd: 32536 rcvadv: 2055868842
```

```
SRTT: 206 ms, RTTO: 300 ms, RTV: 59 ms, KRTT: 0 ms
minRTT: 10 ms, maxRTT: 230 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: 30, connect retry interval: 30 secs
```

```
State flags: none
Feature flags: Win Scale, Nagle
Request flags: Win Scale
```

Datagrams (in bytes): MSS 500, peer MSS 1460, min MSS 500, max MSS 1460

```
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
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
```

```
PDU information:
#PDU's in buffer: 0
FIB Lookup Cache: IFH: 0x200 PD ctx: size: 0 data:
Num Labels: 0 Label Stack:
```

Originating Prefixes With AiGP: Example

The following is a sample configuration for originating prefixes with the AiGP metric attribute:

```
route-policy aigp-policy
  set aigp-metric 4
  set aigp-metric igp-cost
end-policy
!
router bgp 100
  address-family ipv4 unicast
    network 10.2.3.4/24 route-policy aigp-policy
    redistribute ospf ospf metric 4 route-policy aigp-policy
  !
!
end
```

BGP Accept Own Configuration: Example

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

```
router bgp 100
  neighbor 45.1.1.1
    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 45.1.1.1
    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
!

```

BGP Unequal Cost Recursive Load Balancing: Example

This is a sample configuration for unequal cost recursive load balancing:

```

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
  !
!
end

```

Flow-tag propagation

The flow-tag propagation feature enables you to establish a co-relation between route-policies and user-policies. Flow-tag propagation using BGP allows user-side traffic-steering based on routing attributes such as, AS number, prefix lists, community strings and extended communities. Flow-tag is a logical numeric identifier that is distributed through RIB as one of the routing attribute of FIB entry in the FIB lookup table. A flow-tag is instantiated using the 'set' operation from RPL and is referenced in the C3PL PBR policy, where it is associated with actions (policy-rules) against the flow-tag value.

You can use flow-tag propagation to:

- Classify traffic based on destination IP addresses (using the Community number) or based on prefixes (using Community number or AS number).
- Select a TE-group that matches the cost of the path to reach a service-edge based on customer site service level agreements (SLA).
- Apply traffic policy (TE-group selection) for specific customers based on SLA with its clients.
- Divert traffic to application or cache server.

Restrictions for Flow-Tag Propagation

Some restrictions are placed with regard to using Quality-of-service Policy Propagation Using Border Gateway Protocol (QPPB) and flow-tag feature together. These include:

- A route-policy can have either 'set qos-group' or 'set flow-tag,' but not both for a prefix-set.
- Route policy for qos-group and route policy flow-tag cannot have overlapping routes. The QPPB and flow tag features can coexist (on same as well as on different interfaces) as long as the route policy used by them do not have any overlapping route.
- Mixing usage of qos-group and flow-tag in route-policy and policy-map is not recommended.

Configuring Destination-Based Flow-Tag Propagation

The destination-based flow tag feature allows you to match packets based on the flow-tag assigned to the destination address of the incoming packets. Once matched, you can then apply any supported PBR action on this policy.



Note You will not be able to enable both QPPB and flow tag features simultaneously on an interface.

Configuration

Use the following sample configuration to configure destination-based flow-tag propagation.

```
/* Configure a route policy for flow-tag propagation */
Router(config)# prefix-set FLOWTAG36
Router(config-pfx)# 10.1.30.0/24
Router(config-pfx)# end-set
Router(config)# prefix-set FLOWTAG38
Router(config-pfx)# 10.1.40.0/24
Router(config-pfx)# end-set

Router(config)# route-policy SETFLOWTAG
Router(config-rpl)# if destination in FLOWTAG36 then set flow-tag 36 endif
Router(config-rpl)# if destination in FLOWTAG38 then set flow-tag 38 endif
Router(config-rpl)# end-policy
Router(config)# commit
Tue Apr 3 15:10:07.223 IST

/* Configure the class map and policy map for flow-tag propagation */
Router(config)# class-map type traffic match-any FLOWMATCH36
Router(config-cmap)# match flow-tag 36
Router(config-cmap)# end-class-map

Router(config)# class-map type traffic match-any FLOWMATCH38
Router(config-cmap)# match flow-tag 38
Router(config-cmap)# end-class-map

Router(config)# policy-map type pbr FLOWMATCH
Router(config-pmap)# class type traffic FLOWMATCH36
Router(config-pmap-c)# redirect ipv4 nexthop 20.20.20.1
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic FLOWMATCH38
Router(config-pmap-c)# drop
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic DEFAULT
Router(config-pmap-c)# exit
Router(config-pmap)# end-policy-map

/* Configure BGP with flow-tag propagation */
```

```

Router(config)# router bgp 10
Router(config-bgp)# bgp router-id 1.1.1.1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy SETFLOWTAG
Router(config-bgp-af)# redistribute static
Router(config-bgp-af)# bgp attribute-download
Router(config-bgp-af)# redistribute connected
Router(config-bgp-af)# exit

Router(config-bgp)# neighbor 20.20.20.1/24
Router(config-bgp-nbr)# remote-as 20
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy BGPIN in
Router(config-bgp-nbr-af)# route-policy BGPOUT out
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit

Router(config)# route-policy BGPIN
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config)# route-policy BGPOUT
Router(config-rpl)# pass
Router(config-rpl)# end-policy

/* Enter the interface configuration mode and enable flow tag on an interface. */
Router(config)# interface FourHundredGige 0/0/0/0
Router(config-if)# ipv4 address 10.10.10.1 255.255.255.0
Router(config-if)# service-policy type pbr input FLOWMATCH
Router(config-if)# no shut

/* Commit the configuration */
Router(config-if)# commit
Mon Mar 19 07:59:01.081 IST
RP/0/0/CPU0:Mar 19 07:59:01.537 : ifmgr[403]: %PKT_INFRA-LINK-3-UPDOWN : Interface
FourHundredGige0/1/0/0, changed state to Down
RP/0/0/CPU0:Mar 19 07:59:01.619 : ifmgr[403]: %PKT_INFRA-LINK-3-UPDOWN : Interface
FourHundredGige0/2/0/0, changed state to Up

/* Validate the configuraton */
Router(config)# do show run
Mon Mar 19 08:03:31.106 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Mon Mar 19 08:02:55 2018 by UNKNOWN
...
class-map type traffic match-any FLOWMATCH36
  match flow-tag 36
end-class-map
!
!
class-map type traffic match-any FLOWMATCH40
  match flow-tag 40
end-class-map
!
policy-map type pbr FLOWMATCH
  class type traffic FLOWMATCH36
    transmit
  !
  class type traffic FLOWMATCH40
    transmit
  !
  class type traffic class-default

```

```

!
end-policy-map
!
interface FourHundredGige0/1/0/0
  ipv4 forwarding-enable
  ipv6 address 2000::2/64
!
interface FourHundredGige0/2/0/0
  service-policy type pbr input FLOWMATCH
  ipv4 address 10.10.10.1 255.255.255.0
!
interface FourHundredGige0/3/0/0
  ipv4 forwarding-enable
  ipv6 address 3000::2/64
!
...
!
prefix-set FLOWTAG36
  10.1.30.0/24
end-set
!
prefix-set FLOWTAG40
  10.1.40.0/24
end-set
!
route-policy SETFLOWTAG
  if destination in FLOWTAG36 then
    set flow-tag 36
  endif
  if destination in FLOWTAG40 then
    set flow-tag 40
  endif
end-policy
!
!
router bgp 10
  bgp router-id 1.1.1.1
  address-family ipv4 unicast
  table-policy SETFLOWTAG
  redistribute static
  bgp attribute-download
  redistribute connected
!
neighbor 20.20.20.1/24
  remote-as 20
  address-family ipv4 unicast
  route-policy BGPIN in
  route-policy BGPOUT out
!
route-policy BGPIN
  pass
end-policy
route-policy BGPOUT
  pass
end-policy
!

```

You have successfully configured destination-based flow-tag propagation.

Configure Software to Store Updates from Neighbor

Perform this task to configure the software to store updates received from a neighbor.

The **soft-reconfiguration inbound** command causes a route refresh request to be sent to the neighbor if the neighbor is route refresh capable. If the neighbor is not route refresh capable, the neighbor must be reset to relearn received routes using the **clear bgp soft** command.

**Note**

Storing updates from a neighbor works only if either the neighbor is route refresh capable or the **soft-reconfiguration inbound** command is configured. Even if the neighbor is route refresh capable and the **soft-reconfiguration inbound** command is configured, the original routes are not stored unless the **always** option is used with the command. The original routes can be easily retrieved with a route refresh request. Route refresh sends a request to the peer to resend its routing information. The **soft-reconfiguration inbound** command stores all paths received from the peer in an unmodified form and refers to these stored paths during the clear. Soft reconfiguration is memory intensive.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **address-family** { **ipv4** | **ipv6** } **unicast**
5. **soft-reconfiguration inbound** [**always**]
6. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

Step 1 **configure****Example:**

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number***Example:**

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address***Example:**

```
Router(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 4 **address-family { ipv4 | ipv6 } unicast**

Example:

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 5 **soft-reconfiguration inbound [always]**

Example:

```
Router(config-bgp-nbr-af)# soft-reconfiguration inbound always
```

Configures the software to store updates received from a specified neighbor. Soft reconfiguration inbound causes the software to store the original unmodified route in addition to a route that is modified or filtered. This allows a “soft clear” to be performed after the inbound policy is changed.

Soft reconfiguration enables the software to store the incoming updates before apply policy if route refresh is not supported by the peer (otherwise a copy of the update is not stored). The **always** keyword forces the software to store a copy even when route refresh is supported by the peer.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP Route Dampening

Perform this task to configure and monitor BGP route dampening.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family { ipv4 | ipv6 } unicast**
4. **bgp dampening** [*half-life* [*reuse suppress max-suppress-time*]] | **route-policy** *route-policy-name*]
5. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

Step 1 **configure****Example:**

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp** *as-number***Example:**

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family** { **ipv4** | **ipv6** } **unicast****Example:**

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

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 4 **bgp dampening** [*half-life* [*reuse suppress max-suppress-time*]] | **route-policy** *route-policy-name*]**Example:**

```
Router(config-bgp-af)# bgp dampening 30 1500 10000 120
```

Configures BGP dampening for the specified address family.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
 - **No** —Exits the configuration session without committing the configuration changes.
 - **Cancel** —Remains in the configuration session, without committing the configuration changes.
-

Apply Policy When Updating Routing Table

The table policy feature in BGP allows you to configure traffic index values on routes as they are installed in the global routing table. This feature is enabled using the table-policy command and supports the BGP policy accounting feature. Table policy also provides the ability to drop routes from the RIB based on match criteria. This feature can be useful in certain applications and should be used with caution as it can easily create a routing ‘black hole’ where BGP advertises routes to neighbors that BGP does not install in its global routing table and forwarding table.

Perform this task to apply a routing policy to routes being installed into the routing table.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp *as-number***

Example:

```
Router(config)# router bgp 120.6
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family { *ipv4* | *ipv6* } unicast**

Example:

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

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 4 **table-policy *policy-name***

Example:

```
Router(config-bgp-af)# table-policy tbl-plcy-A
```

Applies the specified policy to routes being installed into the routing table.

Step 5 Use the **commit** or **end** command.

commit — Saves the configuration changes and remains within the configuration session.

end — Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.

- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Applying routing policy: Example

In the following example, for an eBGP neighbor, if all routes should be accepted and advertised with no modifications, a simple pass-all policy is configured:

```
Router(config)# route-policy pass-all
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config)# commit
```

Use the **route-policy (BGP)** command in the neighbor address-family configuration mode to apply the pass-all policy to a neighbor. The following example shows how to allow all IPv4 unicast routes to be received from neighbor 192.168.40.42 and advertise all IPv4 unicast routes back to it:

```
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
```

Use the **show bgp summary** command to display eBGP neighbors that do not have both an inbound and outbound policy for every active address family. In the following example, such eBGP neighbors are indicated in the output with an exclamation (!) mark:

```
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
```

Configure BGP Route Filtering by Route Policy

Perform this task to configure BGP routing filtering by route policy.

Procedure

Step 1 **configure**

Step 2 **route-policy** *name*

Example:

```
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
```

(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.

Step 3 **end-policy**

Example:

```
Router(config-rpl)# end-policy
```

(Optional) Ends the definition of a route policy and exits route policy configuration mode.

Step 4 **router bgp** *as-number*

Example:

```
Router(config)# router bgp 120
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 5 **neighbor** *ip-address*

Example:

```
Router(config-bgp)# neighbor 172.168.40.24
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 6 **address-family** { **ipv4** | **ipv6** } **unicast**

Example:

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

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 7 **route-policy** *route-policy-name* { **in** | **out** }

Example:

```
Router(config-bgp-nbr-af)# route-policy drop-as-1234 in
```

Applies the specified policy to inbound routes.

Step 8 **commit**

Configure Destination-based RTBH Filtering

RTBH is implemented by defining a route policy (RPL) to discard undesirable traffic at next-hop using **set next-hop discard** command.

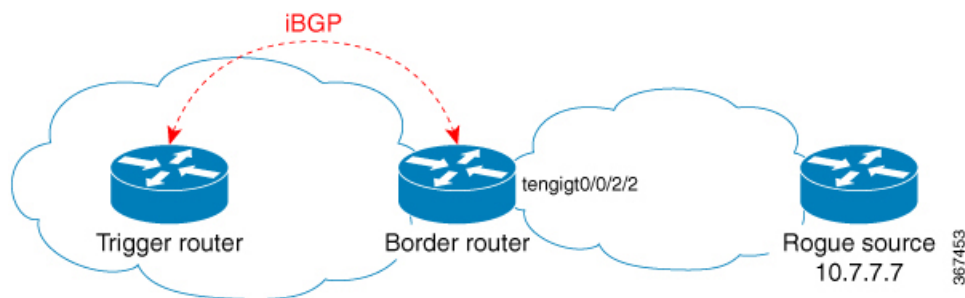
RTBH filtering sets the next-hop of the victim's prefix to the null interface. The traffic destined to the victim is dropped at the ingress.

The **set next-hop discard** configuration is used in the neighbor inbound policy. When this config is applied to a path, though the primary next-hop is associated with the actual path but the RIB is updated with next-hop set to Null0. Even if the primary received next-hop is unreachable, the RTBH path is considered reachable and will be a candidate in the bestpath selection process. The RTBH path is readvertised to other peers with either the received next-hop or nexthop-self based on normal BGP advertisement rules.

A typical deployment scenario for RTBH filtering would require running internal Border Gateway Protocol (iBGP) at the access and aggregation points and configuring a separate device in the network operations center (NOC) to act as a trigger. The triggering device sends iBGP updates to the edge, that cause undesirable traffic to be forwarded to a null0 interface and dropped.

Consider below topology, where a rogue router is sending traffic to a border router.

Figure 16: Topology to Implement RTBH Filtering



Configurations applied on the Trigger Router

Configure a static route redistribution policy that sets a community on static routes marked with a special tag, and apply it in BGP:

```
route-policy RTBH-trigger
  if tag is 777 then
    set community (1234:4321, no-export) additive
    pass
  else
    pass
```

```

endif
end-policy

router bgp 65001
address-family ipv4 unicast
  redistribute static route-policy RTBH-trigger
!
neighbor 192.168.102.1
remote-as 65001
address-family ipv4 unicast
  route-policy bgp_all in
  route-policy bgp_all out

```

Configure a static route with the special tag for the source prefix that has to be block-holed:

```

router static
address-family ipv4 unicast
10.7.7.7/32 Null0 tag 777

```

Configurations applied on the Border Router

Configure a route policy that matches the community set on the trigger router and configure set next-hop discard:

```

route-policy RTBH
  if community matches-any (1234:4321) then
    set next-hop discard
  else
    pass
  endif
end-policy

```

Apply the route policy on the iBGP peers:

```

router bgp 65001
address-family ipv4 unicast
!
neighbor 192.168.102.2
remote-as 65001
address-family ipv4 unicast
  route-policy RTBH in
  route-policy bgp_all out

```

Resilient Hashing and Flow Auto-Recovery

Table 42: Feature History Table

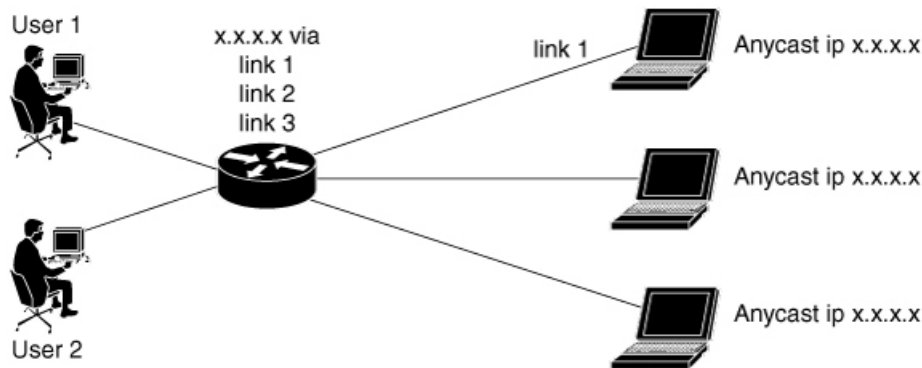
Feature Name	Release	Description
Resilient Hashing and Flow Auto-Recovery	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700 [ASIC:K100])(select variants only*).</p> <p>You can ensure no packet loss and optimal load distribution across available paths by automatically rerouting data 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.</p> <p>*Previously this feature was supported on Q200 and Q100. It is now extended to Cisco 8712-MOD-M routers.</p>

Resilient Hashing and Flow Auto-Recovery feature provides an option to selectively override the default equal cost multipath (ECMP) behavior during a ECMP path failure. This feature enables the redirection of flows through inactive links only and the prevention of all existing flows from being rehashed to a new link. This feature also provides an option to recover a link or a server when it comes back so it can be reused for sessions.

ECMP Path Failure

Prior to the implementation of Resilient Hashing and Flow Auto-Recovery feature, ECMP would load balance the traffic over a number of available paths towards a destination. When one path fails, the traffic gets rehashed over a new set of paths and elects a new next-hop for each path.

Figure 17: ECMP Path Failure

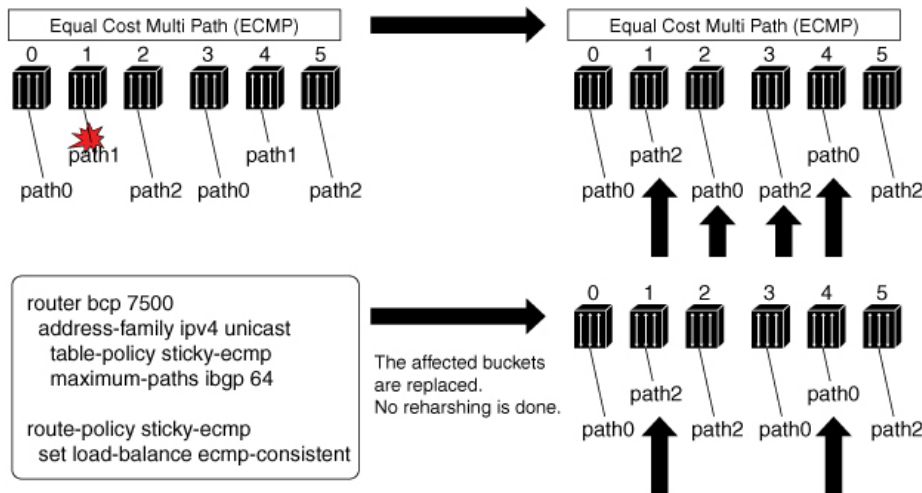


For example, as shown in the figure, among three links link 1, link 2, and link 3, the traffic flow that took link 1 before the failure, takes link 3 after the failure although only link 2 failed.

This traffic flow redistribution does not cause any problem in traditional core networks because the end-to-end connectivity is preserved and the user does not encounter problems from it. However, in data center environments, load balancing due to traffic flow redistribution can cause a problem.

In data center environments where multiple servers are connected through ECMP, the loss of traffic on active link caused by this rehashing resets the TCP session.

Figure 18: Resilient Hashing and Flow Auto-Recovery



The above figure shows how complete rehashing of paths occurs when path 1 fails. However, when Resilient Hashing and Flow Auto-Recovery feature is configured, only the affected buckets are replaced. No rehashing is done. Use an RPL to define prefixes that require resilient hashing and flow auto-recovery. Each prefix has a path list, say for example a prefix 'X' has a path list namely, path 0, path 1, path 2. For example, when path 1 fails and when you have configured Resilient Hashing and Flow Auto-Recovery feature, the new path list becomes (path 0, path 0, and path 2), instead of the default rehash logic, which results (path 0, path 2, and path 0).

When path 1 becomes active, if the Resilient Hashing and Flow Auto-Recovery feature is not configured, no rehashing is done and the path is not utilized until one of the following occurs:

- Addition of new path to ECMP

- Use of **clear route** command.
- Removal of table-policy, commit, addition of table-policy, and commit
- Configuration of **cef consistent-hashing auto-recovery** command

When path 1 becomes active, if the Resilient Hashing and Flow Auto-Recovery feature is configured, the sessions get reshuffled automatically. This causes the sessions, which were moved from the failed path to a new server, to be rehashed back to the original server that became active. Hence, only these sessions are disrupted.

Persistent Loadbalancing

Traditional ECMP or equal cost multipath loadbalances traffic over a number of available paths towards a destination. When one path fails, the traffic gets re-shuffled over the available number of paths. This flow distribution can be a problem in data center loadbalancing.

Persistent Loadbalancing or Sticky ECMP defines a prefix in such a way that it do not rehash flows on existing paths and only replace those bucket assignments of the failed server. The advantage is that the established sessions to servers will not get rehashed.

The following section describes how you can configure persistent load balancing:

```
/*Configure persistent load balancing. */

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
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)#

/* Enable autocovery and hence recover the original hashing state
after failed paths become active. */
Router(config)# cef consistent-hashing auto-recovery

/* Recover to the original hashing state after failed paths come up
and avoid affecting newly formed flows after path failure. */
Router(config)# clear route 192.0.2.0/24
```

Running Configuration

```
/* Configure persistent loadbalancing. */
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
```

Verification

Verify that the path distribution with persistent loadbalancing is configured.

The following show output displays the status of path distribution before a link fails. In this output, three paths are identified with three next hops (10.1/2/3.0.1) through three different GigabitEthernet interfaces.

```
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
```

The following show output displays the status of the path distribution after a link fails. The replacement of bucket 1 with GigabitEthernet 0/0/0/0 and the "*" symbol denotes that this path is a replacement for a failed path.

```
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
```

BGP Selective Multipath

Traditional BGP multipath feature allows a router receiving parallel paths to the same destination to install the multiple paths in the routing table. By default, this multipath feature is applied to all configured peers. BGP selective multipath allows application of the multipath feature only to selected peers.

The BGP router receiving multiple paths is configured with the **maximum-paths ... selective** option. The iBGP/eBGP neighbors sharing multiple paths are configured with the **multipath** option, while being added as neighbors on the BGP router.



Note Use **next-hop-unchanged multipath** command to avoid overwriting next-hop information before advertising multipaths.

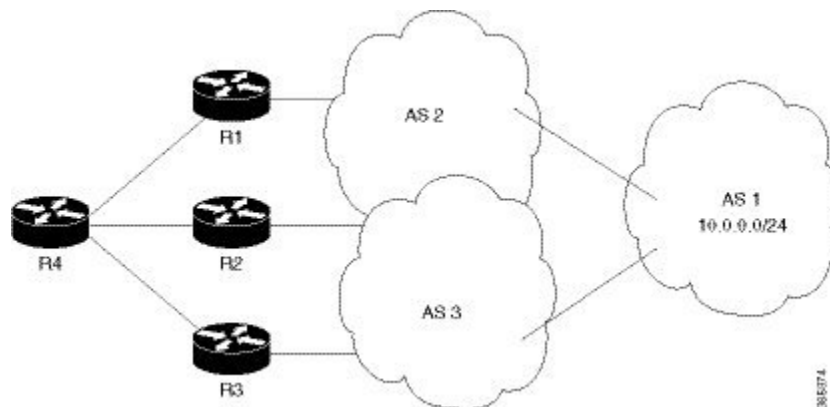
The following behavior is to be noted while using BGP selective multipath:

- BGP selective multipath does not impact best path calculations. A best path is always included in the set of multipaths.
- For VPN prefixes, the PE paths are *always* eligible to be multipaths.

Topology

A sample topology to illustrate the configuration used in this section is shown in the following figure.

Figure 19: BGP Selective Multipath



Router R4 receives parallel paths from Routers R1, R2 and R3 to the same destination. If Routers R1 and R2 are configured as selective multipath neighbors on Router R4, only the parallel paths from these routers are installed in the routing table of Router R4.

Configuration



Note Configure your network topology with iBGP/eBGP running on your routers, before configuring this feature.

To configure BGP selective multipath on Router R4, use the following steps.

1. Configure Router R4 to accept selective multiple paths in your topology.

```

/* To configure selective multipath for iBGP/eBGP
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

/* To configure selective multipath for eiBGP
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

```

2. Configure neighbors for Router R4.

Routers R1 (1.1.1.1) and R2 (2.2.2.2) are configured as neighbors with the **multipath** option.

Router R3 (3.3.3.3) is configured as a neighbor without the **multipath** option, and hence the routes from this router are not eligible to be chosen as multipaths.

```

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

Router(config-bgp-nbr) # neighbor 3.3.3.3
Router(config-bgp-nbr) # address-family ipv4 unicast
Router(config-bgp-nbr-af) # commit

```

You have successfully configured the BGP selective multipath feature.

Remove and Replace Private AS Numbers from AS Path in BGP

Private autonomous system numbers (ASNs) are used by Internet Service Providers (ISPs) and customer networks to conserve globally unique AS numbers. Private AS numbers cannot be used to access the global Internet because they are not unique. AS numbers appear in eBGP AS paths in routing updates. Removing private ASNs from the AS path is necessary if you have been using private ASNs and you want to access the global Internet.

Public AS numbers are assigned by InterNIC and are globally unique. They range from 1 to 64511. Private AS numbers are used to conserve globally unique AS numbers, and they range from 64512 to 65535. Private AS numbers cannot be leaked to a global BGP routing table because they are not unique, and BGP best path calculations require unique AS numbers. Therefore, it might be necessary to remove private AS numbers from an AS path before the routes are propagated to a BGP peer.

External BGP (eBGP) requires that globally unique AS numbers be used when routing to the global Internet. Using private AS numbers (which are not unique) would prevent access to the global Internet. The remove and replace private AS Numbers from AS Path in BGP feature allows routers that belong to a private AS to access the global Internet. A network administrator configures the routers to remove private AS numbers from the AS path contained in outgoing update messages and optionally, to replace those numbers with the ASN of the local router, so that the AS Path length remains unchanged.

The ability to remove and replace private AS numbers from the AS Path is implemented in the following ways:

- The **remove-private-as** command removes private AS numbers from the AS path even if the path contains both public and private ASNs.
- The **remove-private-as** command removes private AS numbers even if the AS path contains only private AS numbers. There is no likelihood of a 0-length AS path because this command can be applied to eBGP peers only, in which case the AS number of the local router is appended to the AS path.
- The **remove-private-as** command removes private AS numbers even if the private ASNs appear before the confederation segments in the AS path.
- The **replace-as** command replaces the private AS numbers being removed from the path with the local AS number, thereby retaining the same AS path length.

The feature can be applied to neighbors per address family (address family configuration mode). Therefore, you can apply the feature for a neighbor in one address family and not on another, affecting update messages on the outbound side for only the address family for which the feature is configured.

Use **show bgp neighbors** and **show bgp update-group** commands to verify that the private AS numbers were removed or replaced.

BGP DMZ Link Bandwidth for Unequal Cost Recursive Load Balancing

Border Gateway Protocol demilitarized zone (BGP DMZ) Link Bandwidth for Unequal Cost Recursive Load Balancing provides support for unequal cost load balancing for recursive prefixes on local node using BGP DMZ Link Bandwidth. The unequal load balance is achieved by using the **dmz-link-bandwidth** command in BGP Neighbor configuration mode and the **bandwidth** command in Interface configuration mode.

BGP Multi-Instance and Multi-AS

Multi-AS BGP enables configuring each instance of a multi-instance BGP with a different AS number. Multi-Instance and Multi-AS BGP provides these capabilities:

- Mechanism to consolidate the services provided by multiple routers using a common routing infrastructure into a single IOS-XR router.
- Mechanism to achieve AF isolation by configuring the different AFs in different BGP instances.
- Means to achieve higher session scale by distributing the overall peering sessions between multiple instances.
- Mechanism to achieve higher prefix scale (especially on a RR) by having different instances carrying different BGP tables.
- Improved BGP convergence under certain scenarios.
- All BGP functionalities including NSR are supported for all the instances.
- The load and commit router-level operations can be performed on previously verified or applied configurations.

Restrictions

- The router supports maximum of 4 BGP instances.
- Each BGP instance needs a unique router-id.
- Only one Address Family can be configured under each BGP instance (VPNv4, VPNv6 and RT-Constrain can be configured under multiple BGP instances).
- IPv4/IPv6 Unicast should be within the same BGP instance in which IPv4/IPv6 Labeled-Unicast is configured.
- IPv4/IPv6 Multicast should be within the same BGP instance in which IPv4/IPv6 Unicast is configured.
- All configuration changes for a single BGP instance can be committed together. However, configuration changes for multiple instances cannot be committed together.
- Cisco recommends that BGP update-source should be unique in the default VRF over all instances while peering with the same remote router.

BGP Prefix Origin Validation Based on RPKI

Table 43: Feature History Table

Feature Name	Release Information	Feature Description
BGP Prefix Origin Validation Based on RPKI	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700) (select variants only*)</p> <p>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.</p> <p>*BGP Prefix Origin Validation Based on RPKI functionality is now supported on Cisco-8712-MOD-M routers.</p>

A BGP route associates an address prefix with a set of autonomous systems (AS) that identify the interdomain path the prefix has traversed in the form of BGP announcements. This set is represented as the AS_PATH attribute in BGP and starts with the AS that originated the prefix.

To help reduce well-known threats against BGP including prefix mis-announcing and monkey-in-the-middle attacks, one of the security requirements is the ability to validate the origination AS of BGP routes. The AS number claiming to originate an address prefix (as derived from the AS_PATH attribute of the BGP route) needs to be verified and authorized by the prefix holder. The Resource Public Key Infrastructure (RPKI) is an approach to build a formally verifiable database of IP addresses and AS numbers as resources. The RPKI is a globally distributed database containing, among other things, information mapping BGP (internet) prefixes to their authorized origin-AS numbers. Routers running BGP can connect to the RPKI to validate the origin-AS of BGP paths.

Configure RPKI Cache-server

Perform this task to configure Resource Public Key Infrastructure (RPKI) cache-server parameters.

Configure the RPKI cache-server parameters in `rpki-server` configuration mode. Use the **rpki server** command in router BGP configuration mode to enter into the `rpki-server` configuration mode

Procedure

Step 1 **configure****Example:**

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **router bgp *as-number*****Example:**

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

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **rpki cache {*host-name* | *ip-address*}****Example:**

```
Router(config-bgp)#rpki server 10.2.3.4
```

Enters `rpki-server` configuration mode and enables configuration of RPKI cache parameters.

Step 4 Use one of these commands:

- **transport ssh port *port_number***
- **transport tcp port *port_number***

Example:

```
Router(config-bgp-rpki-server)#transport ssh port 22
```

Or

```
Router(config-bgp-rpki-server)#transport tcp port 2
```

Specifies a transport method for the RPKI cache.

- **ssh**—Select **ssh** to connect to the RPKI cache using SSH.
- **tcp**—Select **tcp** to connect to the RPKI cache using TCP (unencrypted).
- **port *port_number***—Specify the port number for the RPKI cache transport over TCP and SSH protocols. The port number ranges from 1 to 65535.

Note

- SSH supports custom ports in addition to the default port number 22.
- You can set the transport to either TCP or SSH. Change of transport causes the cache session to flap.

Step 5 (Optional) **username** *user_name***Example:**

```
Router(config-bgp-rpki-server)#username ssh_rpki_cache
```

Specifies a (SSH) username for the RPKI cache-server.

Step 6 (Optional) **password****Example:**

```
Router(config-bgp-rpki-server)#password ssh_rpki_pass
```

Specifies a (SSH) password for the RPKI cache-server.

Note

The “username” and “password” configurations only apply if the SSH method of transport is active.

Step 7 **preference** *preference_value***Example:**

```
Router(config-bgp-rpki-server)#preference 1
```

Specifies a preference value for the RPKI cache. Range for the preference value is 1 to 10. Setting a lower preference value is better.

Step 8 **purge-time** *time***Example:**

```
Router(config-bgp-rpki-server)#purge-time 30
```

Configures the time BGP waits to keep routes from a cache after the cache session drops. Set purge time in seconds. Range for the purge time is 30 to 360 seconds.

Step 9 Use one of these commands.

- **refresh-time** *time*
- **refresh-time off**

Example:

```
Router(config-bgp-rpki-server)#refresh-time 20
```

Or

```
Router(config-bgp-rpki-server)#refresh-time off
```

Configures the time BGP waits in between sending periodic serial queries to the cache. Set refresh-time in seconds. Range for the refresh time is 15 to 3600 seconds.

Configure the **off** option to specify not to send serial-queries periodically.

Step 10 Use one these commands.

- **response-time** *time*
- **response-time off**

Example:

```
Router(config-bgp-rpki-server)#response-time 30
```

Or

```
Router(config-bgp-rpki-server)#response-time off
```

Configures the time BGP waits for a response after sending a serial or reset query. Set response-time in seconds. Range for the response time is 15 to 3600 seconds.

Configure the **off** option to wait indefinitely for a response.

Step 11 shutdown

Example:

```
Router(config-bgp-rpki-server)#shutdown
```

Configures shut down of the RPKI cache.

Step 12 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

BGP Update Message Error Handling

The BGP UPDATE message error handling changes BGP behavior in handling error UPDATE messages to avoid session reset. Based on the approach described in IETF IDR *I-D:draft-ietf-idr-error-handling*, the Cisco IOS XR BGP UPDATE Message Error handling implementation classifies BGP update errors into various categories based on factors such as, severity, likelihood of occurrence of UPDATE errors, or type of attributes. Errors encountered in each category are handled according to the draft. Session reset will be avoided as much as possible during the error handling process. Error handling for some of the categories are controlled by configuration commands to enable or disable the default behavior.

According to the base BGP specification, a BGP speaker that receives an UPDATE message containing a malformed attribute is required to reset the session over which the offending attribute was received. This behavior is undesirable as a session reset would impact not only routes with the offending attribute, but also other valid routes exchanged over the session.

BGP Attribute Filtering

The BGP Attribute Filter feature checks integrity of BGP updates in BGP update messages and optimizes reaction when detecting invalid attributes. BGP Update message contains a list of mandatory and optional attributes. These attributes in the update message include MED, LOCAL_PREF, COMMUNITY etc. In some cases, if the attributes are malformed, there is a need to filter these attributes at the receiving end of the router. The BGP Attribute Filter functionality filters the attributes received in the incoming update message. The attribute filter can also be used to filter any attributes that may potentially cause undesirable behavior on the receiving router.

Some of the BGP updates are malformed due to wrong formatting of attributes such as the network layer reachability information (NLRI) or other fields in the update message. These malformed updates, when

received, causes undesirable behavior on the receiving routers. Such undesirable behavior may be encountered during update message parsing or during re-advertisement of received NLRIs. In such scenarios, its better to filter these corrupted attributes at the receiving end.

BGP Error Handling and Attribute Filtering Syslog Messages

When a router receives a malformed update packet, an `ios_msg` of type `ROUTING-BGP-3-MALFORM_UPDATE` is printed on the console. This is rate limited to 1 message per minute across all neighbors. For malformed packets that result in actions "Discard Attribute" (A5) or "Local Repair" (A6), the `ios_msg` is printed only once per neighbor per action. This is irrespective of the number of malformed updates received since the neighbor last reached an "Established" state.

This is a sample BGP error handling syslog message:

```
%ROUTING-BGP-3-MALFORM_UPDATE : Malformed UPDATE message received from neighbor 13.0.3.50
- message length 90 bytes,
  error flags 0x0000840, action taken "TreatAsWithdraw".
Error details: "Error 0x0000800, Field "Attr-missing", Attribute 1 (Flags 0x00, Length 0),
  Data []"
```

This is a sample 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 40.0.101.1
- message length 173 bytes,
  action taken "DiscardAttr".
Filtering details: "Attribute 16 (Flags 0xc0): Action "DiscardAttr"". NLRIs: [IPv4 Unicast]
88.2.0.0/17
```

This is a sample 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 40.0.101.1
- message length 166 bytes,
  action taken "TreatAsWdr".
Filtering details: "Attribute 4 (Flags 0xc0): Action "TreatAsWdr"". NLRIs: [IPv4 Unicast]
88.2.0.0/17
```

BGP-RIB Feedback Mechanism for Update Generation

The Border Gateway Protocol-Routing Information Base (BGP-RIB) feedback mechanism for update generation feature avoids premature route advertisements and subsequent packet loss in a network. This mechanism ensures that routes are installed locally, before they are advertised to a neighbor.

BGP waits for feedback from RIB indicating that the routes that BGP installed in RIB are installed in forwarding information base (FIB) before BGP sends out updates to the neighbors. RIB uses the the BCDL feedback mechanism to determine which version of the routes have been consumed by FIB, and updates the BGP with that version. BGP will send out updates of only those routes that have versions up to the version that FIB has installed. This selective update ensures that BGP does not send out premature updates resulting in attracting

traffic even before the data plane is programmed after router reload, LC OIR, or flap of a link where an alternate path is made available.

To configure BGP to wait for feedback from RIB indicating that the routes that BGP installed in RIB are installed in FIB, before BGP sends out updates to neighbors, use the **update wait-install** command in router address-family IPv4 or router address-family VPNv4 configuration mode. The **show bgp**, **show bgp neighbors**, and **show bgp process performance-statistics** commands display the information from update wait-install configuration.

Configure BGP Large Communities

BGP communities provide a way to group destinations and apply routing decisions such as acceptance, rejection, preference, or redistribution on a group of destinations using community attributes. BGP community attributes are variable length attributes consisting of a set of one or more 4-byte values which are split into two parts of 16 bits. The higher-order 16 bits represents the AS number and the lower order bits represents a locally defined value assigned by the operator of the AS.

Since the adoption of 4-byte ASNs (RFC6793), the BGP communities attribute can no longer accommodate the 4 byte ASNs as you need more than 4 bytes to encode the 4-byte ASN and an AS specific value that you want to tag with the route. Although BGP extended community permits a 4-byte AS to be encoded as the global administrator field, the local administrator field has only 2-byte of available space. So, 6-byte extended community attribute is also unsuitable. To overcome this limitation, you can configure a 12-byte BGP large community which is an optional attribute that provides the most significant 4-byte value to encode autonomous system number as the global administrator and the remaining two 4-byte assigned numbers to encode the local values.

Similar to BGP communities, routers can apply BGP large communities to BGP routes by using route policy languages (RPL) and other routers can then perform actions based on the community that is attached to the route. The policy language provides sets as a container for groups of values for matching purposes.

When large communities are specified in other commands, they are specified as three non negative decimal integers separated by colons. For example, 1:2:3. Each integer is stored in 32 bits. The possible range for each integer is 0 to 4294967295.

In route-policy statements, each integer in the BGP large community can be replaced by any of the following expressions :

- [x..y] — This expression specifies a range between x and y, inclusive.
- * — This expression stands for any number.
- peeras — This expression is replaced by the AS number of the neighbor from which the community is received or to which the community is sent, as appropriate.
- not-peeras — This expression matches any number other than the peeras.
- private-as — This expression specifies any number in the private ASN range: [64512..65534] and [4200000000..4294967294].

These expressions can be also used in policy-match statements.

IOS regular expression (ios-regex) and DFA style regular expression (dfa-regex) can be used in any of the large-community policy match and delete statements. For example, the IOS regular expression ios-regex `^5:.*:7$` is equivalent to the expression `5:*:7`.

The **send-community-ebgp** command is extended to include BGP large communities. This command is required for the BGP speaker to send large communities to ebgp neighbors.

Restrictions and Guidelines

The following restrictions and guidelines apply for BGP large communities:

- All functionalities of 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.
- The peer-as expression cannot be used in a large-community-set.
- The peer-as expression can only be used in large-community match or delete statements that appear in route policies that are applied at the neighbor-in or neighbor-out attach points.
- The not-peer-as expression cannot be used in a large-community-set or in policy set statements.

Configuration Example: Large Community Set

A large-community set defines a set of large communities. Named large-community sets are used in route-policy match and set statements.

This example shows how to create a named large-community set.

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

Configuration Example: Set Large Community

The following example shows how to set the BGP large community attribute in a route, using the **set large-community** {*large-community-set-name* | *inline-large-community-set* | *parameter*} [**additive**] command. You can specify a named large-community-set or an inline set. The **additive** keyword retains the large communities already present in the route and adds the new set of large communities. However the **additive** keyword does not result in duplicate entries.

If a particular large community is attached to a route and you specify the same large community again with the additive keyword in the set statement, then the specified large community is not added again. The merging operation removes duplicate entries. This also applies to the peer-as keyword.

The peer-as expression in the example is replaced by the AS number of the neighbor from which the BGP large community is received or to which the community is sent, as appropriate.

```
Router(config)# route-policy mordac
Router(config-rpl)# set large-community (1:2:3, peer-as:2:3)
Router(config-rpl)# end-set
Router(config)# large-community-set catbert
Router(config-largecomm)# 1: 2: 3,
Router(config-largecomm)# peer-as:2:3
Router(config-largecomm)# end-set
Router(config)# route-policy wally
Router(config-rpl)# set large-community catbert additive
Router(config-rpl)# end-set
```

In this example, if the route-policy mordac is applied to a neighbor, the ASN of which is 1, then the large community (1:2:3) is set only once.



Note You should configure the **send-community-ebgp** command to send large communities to ebgp neighbors.

Configuration Example: Large Community Matches-any

The following example shows how to configure a route policy to match any element of a large -community set. This is a boolean condition and returns true if any of the large communities in the route match any of the large communities in the match condition.

```
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
```

Configuration Example: Large Community Matches-every

The following example shows how to configure a route policy where every match specification in the statement must be matched by at least one large community in the route.

```
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
```

In this example, routes with these sets of large communities return TRUE:

- (1:1:3, 4:5:10)
- (4:5:3) —This single large community matches both specifications.
- (1:1:3, 4:5:10, 7:6:5)

Routes with the following set of large communities return FALSE:

(1:1:3, 5:5:10)—The specification (4:5:*) is not matched.

Configuration Example: Large Community Matches-within

The following example shows how to configure a route policy to match within a large community set. This is similar to the **large-community matches-any** command but every large community in the route must match at least one match specification. Note that if the route has no large communities, then it matches.

```
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
```

For example, routes with these sets of large communities return TRUE:

- (1:1:3, 4:5:10)
- (4:5:3)

- (1:2:3, 6:6:3, 9:4:3)

Routes with this set of large communities return FALSE:

(1:1:3, 4:5:10, 7:6:5) —The large community (7:6:5) does not match

Configuration Example: Community Matches-within

The following example shows how to configure a route policy to match within the elements of a community set. This command is similar to the **community matches-any** command, but every community in the route must match at least one match specification. If the route has no communities, then it matches.

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

For example, routes with these sets of communities return TRUE:

- (1:3, 5:10)
- (5:3)
- (2:3, 6:3, 4:3)

Routes with this set of communities return FALSE:

(1:3, 5:10, 6:5) —The community (6:5) does not match.

Configuration Example: Large Community Is-empty

The following example shows using the **large-community is-empty** clause to filter routes that do not have the large-community attribute set.

```
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
```

Configuration Example: Attribute Filter Group

The following example shows how to configure and apply the attribute-filter group with large-community attributes for a BGP neighbor. The filter specifies the BGP path attributes and an action to take when BGP update message is received. If an update message is received from the BGP neighbor that contains any of the specified attributes, then the specified action is taken. In this example, the attribute filter named dogbert is created and applied to the BGP neighbor 10.0.1.101. It specifies the large community attribute and the action of discard. That means, if the large community BGP path attribute is received in a BGP UPDATE message from the neighbor 10.0.1.101 then the attribute will be discarded before further processing of the message.

```
Router(config)# router bgp 100
Router(config-bgp)# attribute-filter group dogbert
Router(config-bgp-attrib)# attribute LARGE-COMMUNITY discard
Router(config-bgp-attrib)# 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
```

Configuration Example: Deleting Large Community

The following example shows how to delete specified BGP large-communities from a route policy using the **delete large-community** command.

```
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:*)
```

Verification

This example displays the routes with large-communities given in the **show bgp large-community list-of-large-communities [exact-match]** command. If the optional keyword **exact-match** is used, then the listed routes will contain only the specified large communities. Otherwise, the displayed routes may contain additional large communities.

```
Router:R1# show bgp large-community 1:2:3 5:6:7
Thu Mar 23 14:40:33.597 PDT
BGP router identifier 4.4.4.4, 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 ?
```

This example displays the large community attached to a network using the **show bgp ip-address/prefix-length** command.

```
Router:R4# 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
```

Resetting an eBGP Session Immediately Upon Link Failure

By default, if a link goes down, all BGP sessions of any directly adjacent external peers are immediately reset. Use the **bgp fast-external-fallover disable** command to disable automatic resetting. Turn the automatic reset back on using the **no bgp fast-external-fallover disable** command.

eBGP sessions flap when the node reaches 3500 eBGP sessions with BGP timer values set as 10 and 30. To support more than 3500 eBGP sessions, increase the packet rate by using the **lpts pifib hardware police location location-id** command. Following is a sample configuration to increase the eBGP sessions:

```
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
```

User-Defined Martian Address Check

When you configure BGP on a Cisco 8000 Series Router, you can prevent routers from accessing certain sites with certain IP address prefixes. These routers drop packets from such IP addresses, and such IP addresses are known as 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 command **default-martian-check disable**. These sites are sites with certain IPv4 and IPv6 prefixes as follows:

- IPv4 address prefixes
 - 0.0.0.0/8
 - 127.0.0.0/8
 - 224.0.0.0/4
- IPv6 address prefixes
 - ::
 - ::0002 - ::ffff
 - ::ffff:a.b.c.d
 - fe80:xxxx
 - ffxx:xxxx

Restrictions

Routers with OSPF or IS-IS Protocols cannot access these sites even by having the **default-martian-check disable** command configured.

Configuration Example

To allow routes from Martian addresses, use the following steps:

1. Enter BGP IPv4 or BGP IPv6 address-family configuration mode.
2. Configure the address-family modifier as a unicast address.
3. Disable the Martian address check.

Configuration

```
/* Enter BGP IPv4 or BGP IPv6 address-family configuration mode. */
Router# configure
Router(config)# router bgp 100

/* Configure the address-family modifier as unicast. */
Router(config-bgp)# address-family ipv4 unicast

/* Disable the martian address check. */
Router(config-bgp-af)# default-martian-check disable
Router(config-bgp-af)# commit
```

Verification

To verify if you have enabled or disabled a Martian address check, you can use the **show bgp ipv4 unicast** command or **show bgp ipv6 unicast** command:

```
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      100        0         i
* i192:1::/112    1.1.1.1            0         100        0 ?
*>i               1:1:1:1:1:1:1      0         100        0 ?
* i11:1:1:1:1:1  1.1.1.1            2         100        0 ?
*>i               1:1:1:1:1:1:1      2         100        0 ?
```

BGP Labeled Unicast MPLS IP POP Support

Table 44: 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.

Consider a topology in which client A that connects to PE1 sends IPv4 unicast traffic to destination client B with IP address 103.101.1.1/24. Client B is connected to PE3. PE1 learns the destination IP address from BGP label unicast with implicit NULL that PE3 advertises. PE1 only adds the transport LDP label on top of the IPV4 unicast packet. PE1 then forwards the packet to the core.

Configuration Example

Configure the following steps on PE1.

```
Router(config)# hw-module profile bgplu enable
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
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
Router(config-bgp)# neighbor 10.3.3.3
Router(config-bgp)# remote-as 200
Router(config-bgp)# update-source Loopback0
Router(config-bgp)# address-family ipv4 labeled-unicast
Router(config-bgp)# next-hop self

/* Note: Restart the router for the hw-module command configuration to take effect. */
```

Configure the following steps on PE3.

```
Router(config)# hw-module profile bgplu enable
Router(config)# router bgp 200
Router(config-bgp)# nsr
Router(config-bgp)# bgp router-id 192.168.70.25
Router(config-bgp)# bgp graceful-restart
Router(config-bgp)# ibgp policy out enforce-modifications
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# maximum-paths ibgp 8
Router(config-bgp-af)# network 103.101.1.0/24
Router(config-bgp-af)# network 103.101.2.0/24
Router(config-bgp-af)# allocate-label all
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.1.1.1
Router(config-bgp)# remote-as 1
Router(config-bgp)# update-source Loopback0
Router(config-bgp)# address-family ipv4 labeled-unicast
Router(config-bgp)# next-hop self

/* Note: Restart the router for the hw-module command configuration to take effect. */
```

Running Configuration

Following is the running configuration on PE1.

```
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

```

Following is the running configuration on PE3.

```

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

```

Verification

The following example verifies the feature.

```

Router# show bgp ipv4 unicast labels
      Network      Next Hop      Rcvd Label      Local Label
*>i103.101.1.0/24  3.3.3.3          3              24006

```

BGP eBGP Security GTSM

Table 45: Feature History Table

Feature Name	Release Information	Feature Description
BGP-eBGP Security GTSM	Release 7.3.1	<p>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.</p> <p>New command introduced:</p> <ul style="list-style-type: none"> • ttl-security

GTSM is based on the fact that a vast majority of protocol peerings are established between routers that are adjacent. Hence most protocol peerings occur either directly between connected interfaces or, between loopback and loopback, with static routes to loopback interfaces. As TTL spoofing is almost impossible to occur, a mechanism based on an expected TTL value provides a simple and robust protection from infrastructure attacks that are based on forged protocol packets coming from outside the network. GTSM mechanism applies to both TTL, that is IPv4, and Hop Limit, that is IPv6.

You can configure the BGP-eBGP Security GTSM feature using the **ttl-security** command.

You can configure **ttl-security** command to enable GTSM for eBGP neighbors. When you configure **ttl-security** command for an eBGP neighbor that is directly adjacent, the router accepts only IP packets with a TTL count that is equal to the maximum TTL value.

Configure **ttl-security** command to secure the eBGP session only in the ingress direction. In the egress direction, the feature causes packets to be sent only with the maximum TTL value so that the BGP neighbor also verifies the TTL value of ingress packets. This feature enables BGP to establish or maintain a session only if the TTL value in the IP packet header is equal to the maximum TTL value. If the value is less than the maximum TTL value, the packet is discarded. In this case, an Internet Control Message Protocol (ICMP) message is not generated. This functionality is enable so that a response to a forged packet is not required.

Configuration Example

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 7000::1
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
```

Running Configuration

```
router bgp 100
 neighbor 7000::1
  remote-as 200
  ebgp-multihop 255
  ttl-security
  address-family ipv6 unicast
  multipath
  route-policy PASS_ALL in
  route-policy PASS_ALL out
```

Steering of BGP Control-Plane Traffic over IP Path

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 (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	<p>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.</p> <p>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.</p> <p>The feature introduces these changes:</p> <p>CLI:</p> <p>New Commands:</p> <ul style="list-style-type: none"> • table ip-only activate vrf • tcp ip-only-preferred <p>Modified Commands:</p> <ul style="list-style-type: none"> • The distribute-list command is modified with a new ip-only keyword. <p>YANG Data Models: New XPathS for</p> <ul style="list-style-type: none"> • 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 <p>(see GitHub, YANG Data Models Navigator)</p>

For an underlay network, the transport LSP is set up using a Multiprotocol Label Switching (MPLS) protocol such as Segment-Routing MPLS, Label Distribution Protocol (LDP), or Service Layer API. When a transport LSP is installed for traffic forwarding, it becomes the standard route for all traffic going toward the underlay destination. This includes labeled traffic, IP traffic, and BGP control-plane traffic. Routing of BGP control-plane traffic through MPLS LSP can introduce complexity and risk, and cause network instability or fluctuations.

The Steering of BGP Control-Plane Traffic over IP Path feature allows you to choose IP-only transport path for the BGP control-plane traffic. This path is set up by the IS-IS protocol. With this feature, MPLS routes are used to determine the next-hop for BGP, while IP-only routes are used to forward traffic to the specific destination set by IGP.

Before enabling the feature, you must create a new VRF to handle IP-only routing tables. When the feature is enabled through configuration, IS-IS creates an IP-only route entry in the Routing Information Base (RIB). RIB then downloads this entry to the Forwarding Information Base (FIB) in the newly created VRF. This VRF provides a separate topology for resolving the locally generated BGP control-plane traffic.

Configure Steering of BGP Control Plane Traffic through an IP-only Path

Presently, IP-only preference for TCP session packets can be enabled only in BGP neighbor group, and no specific configuration is required in IS-IS. You can control the routes in IP-only table by using the **distribute-list** command.

Configuration Example

This example shows how to configure steering of BGP control-plane traffic through an IP-only path.

```
/* Configure ip-only VRF.*/
Router# config
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

/* Configure ip-only steering under RIB configuration mode.*/
Router# config
Router(config)# router rib
Router(config-rib)# table ip-only activate vrf ip_only

/* Configure BGP neighbor to use ip-only steering.*/
Router(config)# router bgp 140
Router(config-bgp)# neighbor-group ip-only
Router(config-bgp-nbrgrp)# remote-as 100
Router(config-bgp-nbrgrp)# update-source Loopback99
Router(config-bgp-nbrgrp)# tcp ip-only-preferred

/* Configure ip-only steering under prefix-list that allows host-routes only.*/
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

/* Configure ip-only steering under route-policy.*/
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
```



Note **tcp ip-only-preferred** command implies a preferred option. If an IP-only path isn't available, the system automatically uses any available path without interrupting the BGP session.

Running Configuration

```
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
  !
  !
```

Verification

Use the following commands to verify if the feature is enabled.

```
Router# show running-config router rib
Wed Mar 27 06:39:01.233 UTC
router rib
  table ip-only activate vrf ip_only
  !
```

```
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

```

```
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

Use the following command to display the IS-IS IP-only local RIB entries:

```
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

```

Use the following command to verify if BGP is setting IP-only option, and to view the table TCP details for that neighbor:

```
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	Wakeup	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_BLOCKSNDB
 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

Use the following command to display the number of IP-only packets per neighbor.

```
Router# show tcp statistics pcb 0x00007f733000d618 location 0/rP1/CPU0
Wed Mar 27 06:46:52.566 UTC
```

```
=====
Statistics for PCB 0x7flca0008550, 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
```

128 Multipath ECMP

Table 47: Feature History Table

Feature Name	Release Information	Feature Description
128-way ECMP	Release 7.3.1	This feature enables the router to support up to 128 BGP and 64 IGP parallel multipaths to a destination.

Equal-cost multi-path routing (ECMP) is a routing strategy where next-hop packet forwarding to a single destination can occur over multiple “best paths” which tie for top place in routing metric calculations. Multipath routing can be used in conjunction with most routing protocols, since it is a per-hop decision that is limited to a single router. It potentially offers substantial increases in bandwidth by load-balancing traffic over multiple paths. 128-path ECMP feature enables the router to support up to 128 parallel paths to a destination.

Cisco 8000 Series Routers supports configuration of up to 128 ECMP next hops for BGP in IPv4 and IPv6. Support for a maximum of 128 path ECMP is available in the global table for BGP, particularly in iBGP and eBGP prefixes.

Restrictions

- VRF and global tables in IGP maximum paths support 64-path ECMP and a maximum of 128 paths are supported with back up.

- BGP and IGP in UCMP support 64-path ECMP.
- LDP gets only 64-ECMP support from RIB, hence LSD gets only a maximum of 64 ECMPs from all clients.

Configure 128-Multipath BGP ECMP

- Configure 128-Multipath ECMP in iBGP
- Configure 128-Multipath ECMP in eBGP
- Configure 128-Multipath ECMP in eiBGP



Note You can configure eBGP and iBGP multipath together. However, if eBGP configuration is already present, then you cannot configure iBGP or eBGP.

Configuration

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
```

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
```

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

The following is the running configuration for 128-multipath ECMP in iBGP.

```
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
```

The following is the running configuration for 128-multipath ECMP in eBGP.

```
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
```

The following is the running configuration for 128-multipath ECMP in eiBGP.

```
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
```

Verification

Verify the BGP multipath marking.

```
Router# show bgp 192.0.2.254/24
Fri Aug 14 13:59:45.190 UTC
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
```

Verify the BGP multipath marking in FIB.

```
Router# show cef 192.0.2.254/24
Fri Aug 14 14:09:50.987 UTC
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]
```

Verify the BGP multipath marking in RIB.

```
Routing# show route 192.0.2.254/24
Fri Aug 14 14:11:27.403 UTC

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
```

Verify the total number of IPv4 ECMP paths

```
Router# show bgp ipv4 unicast 191.1.0.0/24 | i multipath | utility wc -l
Tue Jun 30 12:40:52.432 UTC
128
```

Verify the total number of IPv6 ECMP paths.

```
Router# show bgp ipv6 unicast 191:1::/64 | i multipath | utility wc -l
Tue Jun 30 12:42:28.893 UTC
128
```

Verify the IPv4 128-multipath ECMP.

```
Router# show route ipv4 191.1.0.0/24 | i multipath | utility wc -l
Tue Jun 30 12:40:53.253 UTC
128
Router# show cef ipv4 191.1.0.0/24 | i multipath | utility wc -l
Tue Jun 30 12:40:53.864 UTC
128
```

Verify the IPv6 128-multipath ECMP.

```
Router# show route ipv6 191:1::/64 | i multi | utility wc -l
Tue Jun 30 12:42:29.709 UTC
```

```

128
Router# show cef ipv6 191:1::/64 | i multipath | utility wc -l
Tue Jun 30 12:42:30.332 UTC
128

```

Verify the ECMP capabilities of the platform.

```

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 Multipath ECMP

Table 48: Feature History Table

Feature Name	Release Information	Feature Description
256-way ECMP	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8200, 8700); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])</p> <p>You can now configure up to 256 ECMP next hops for BGP in both IPv4 and IPv6, optimizing network bandwidth by load-balancing traffic across parallel paths.</p>

Equal-cost multi-path routing (ECMP) is a routing strategy where next-hop packet forwarding to a single destination can occur over multiple “best paths” which tie for top place in routing metric calculations. Multipath routing can be used in conjunction with most routing protocols, since it is a per-hop decision that is limited to a single router. It potentially offers substantial increases in bandwidth by load-balancing traffic over multiple paths. 256-path ECMP feature enables the router to support up to 256 parallel paths to a destination.

Cisco 8000 Series Routers supports configuration of up to 256 ECMP next hops for BGP in IPv4 and IPv6. Support for a maximum of 256 path ECMP is available in the global table for BGP, particularly in iBGP and eBGP prefixes.

Restrictions

These restrictions apply to the 256-way ECMP feature.

- BGP and IGP in UCMP support 256-path ECMP for IP, and 64-path ECMP for MPLS.
- LDP gets only 64-ECMP support from RIB, hence LSD gets only a maximum of 64 ECMPs from all clients.

Configure 256-Multipath BGP ECMP

Perform the following steps:

- Configure 256-multipath ECMP in iBGP
- Configure 256-multipath ECMP in eBGP
- Configure 256-multipath ECMP in eiBGP



Note You can configure eBGP and iBGP multipath together. However, if eiBGP multipath configuration is already present, you cannot configure iBGP or eBGP multipath.

Procedure

Step 1 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
```

Step 2 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
```

Step 3 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 4 Verify the BGP multipath marking.

Example:

```

Router# show bgp 10.0.2.254/24
Fri Aug 14 13:59:45.190 UTC
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 5 Verify the BGP multipath marking in FIB, and RIB.

Example:

BGP multipath marking in FIB

```

Router# show cef 10.0.2.254/24
Fri Aug 14 14:09:50.987 UTC
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]

```

Example:

BGP multipath marking in RIB

```

Router# show route 10.0.2.254/24
Fri Aug 14 14:11:27.403 UTC

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 6 Verify the total number of IPv4, IPv6 ECMP paths.

Example:

```

Router# show bgp ipv4 unicast 10.1.0.0/24 | i multipath | utility wc -l
Tue Jun 30 12:40:52.432 UTC
256

Router# show bgp ipv6 unicast 2001:DB8::1 | i multipath | utility wc -l
Tue Jun 30 12:42:28.893 UTC
256

```

Step 7 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

```

Configure 1024 Equal Cost Multi-Path or 256 Unequal Cost Multi-Path using Hierarchical Load Balancing

Table 49: Feature History Table

Feature Name	Release Information	Feature 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 1024 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.

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 1024 ECMPs or 256 UCMPs by using hierarchical-load-balancing model. FIB does the route split and path grouping based on your configuration, such as AS number of each path, or based on the group-size, and also on your scenario.

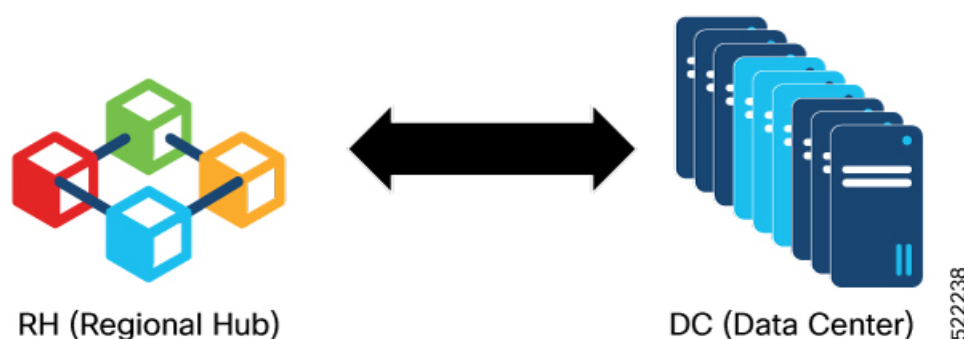
As a prerequisite, Cisco Express Forwarding (CEF) must be enabled on all participating routers, because load balancing requires CEF. In unequal cost multipath (UCMP) load-balancing, a weight is associated with each next hop and traffic is distributed across the next hops in proportion to their weight. In ECMP, the route to a destination has multiple next hops and traffic is equally distributed.

The support until now was only 128 ECMPs. Starting from Release 7.3.3, the support is extended to 1024 ECMPs or 256 UCMPs. So, at a time, you can configure either 1024 ECMPs or 256 UCMPs.

- Router-specific FIB needs to group the paths to two levels.
- The FIB splits routes based on the configured group size.
- BGP downloads the neighbor AS for each path along with other information to RIB; RIB then sends to FIB so that FIB does the regrouping.

By enabling hierarchical load-balancing ECMP group-size of 32 data centers, each data center has about 32 different interface connections. So, in total, consider $32 \times 32 = 1024$ ECMPs.

Figure 20: Sample Topology of Equal Cost Multi-Path



The sample topology shows the Regional Hub connected with 32 data centers with eBGP interface peering. Each data center has 32 interface connections, which add up to a total of 1024 ECMPs.

By enabling hierarchical load-balancing UCMP group-size of 256 within a data center, and scale 256 UCMP eBGP paths on top-of-rack switching with weights.

Restrictions for Hierarchical Load Balancing

- The hierarchical-load-balancing model works only for IP paths, which is enhance eBGP interface peering scenario.
- This model doesn't apply, if any path that has MPLS label or SRv6 information because the information makes CEF unable to convert the flat load-balancing to hierarchical-load-balancing model.

Enabling Hierarchical Load Balancing

To enable hierarchical load balancing with 1024 paths, use the **cef hierarchical-load-balancing** command. For ECMP mode, the FIB splits or groups the paths according to path attribute of the AS number. Paths to the same destination must be configured on the same AS. Ensure, however, that the maximum number of remote AS numbers are less than 128 due to hardware limitations.

For ECMP:

- If the paths are less than the **min-path**, FIB uses native forwarding mode.

- If the paths are greater than or equal to the **min-path**, FIB converts the forwarding chain to hierarchical forwarding.

This is the running configuration for ECMP:

```
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 for the **cef hierarchical-load-balancing ecmp min-paths** command to take effect.

Configuring Group Size and Route Policy

To control the traffic on UCMP link, you can define the extended community bandwidth routing policy with the weights. For example, the ratio is **AS number: Weight number** is applied on the egress of BGP.

Group size is used for FIB to split or group paths to hierarchical forwarding mode. The group size value must be within the hardware limitation of 64 (IGP level load balancing). The recommended group size is 64 if all the paths are ECMP. However, but if paths have weight attribute, then the recommended group size is only 32.

This is the running configuration to enable hierarchical forwarding:

```
cef hierarchical-load-balancing ucmp group-size 32

route-policy BW1
  set extcommunity bandwidth (800:10000)
end-policy
!
route-policy BW2
  set extcommunity bandwidth (800:20000)
end-policy
!
route-policy BW3
  set extcommunity bandwidth (800:40000)
end-policy
!
route-policy BW4
  set extcommunity bandwidth (800:80000)
end-policy
!
```



Note Reload the router for the **cef hierarchical-load-balancing ucmp group-size** command to take effect.

Verifying CEF Configuration

This show output displays the bgp ipv4 unicast prefix status:

```
Router1#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
```

This show output displays the CEF configuration status for ECMP:

```
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 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
```

This show output displays the CEF configuration status for UCMP:

```
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

```

BGP Slow Peer

Table 50: Feature History Table

Feature Name	Release	Description
BGP Slow Peer	Release 7.9.1	<p>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.</p> <p>This feature introduces the following commands:</p> <ul style="list-style-type: none"> • slow peer (BGP neighbor address-family configuration) • slow peer (BGP neighbor address-family configuration) <p>This feature modifies the following commands:</p> <ul style="list-style-type: none"> • 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

BGP neighbors are grouped together based on some common criteria such as neighbor address-family configuration and processing of same update messages, used to optimize performance when generating updates. For each group, the messages are formatted and then transmitted to all the members of the group. When all members of the group have acknowledged receipt of the messages, only then the messages are deleted.

One of the main drawbacks of grouping neighbors for an update generation is the impact of slow peers on the remaining peers in the group. For example, Cisco IOS XR BGP has limits on the update messages per process, per address-family, and per sub-group. The default sub-group message limit is 32 Mbytes. If one or more BGP peers in the sub-group are extremely slow to process messages, those messages must be kept in the queue until acknowledged by all sub-group peers, and if the sub-group message limit is reached, all neighbors in the sub-group must wait until the slowest peer catches up. This slows all members of the sub-group. Slow peer handling mitigates the impacts of slow peer on other peers of the sub-group.

Cisco IOS XR BGP update generation is per address-family, a peer refers to a neighbor address-family. So peers in a sub-group are grouping of neighbors for a particular address-family.

A BGP peer can be slow for many reasons. Below are a few reasons why a peer can be slow:

- Processor is busy handling other tasks and cannot process the updates on time
- Connected over slow bandwidth links
- Temporary network congestion

Slow peer handling is important when routes are constantly changing over a long period of time. It is important to clean up stale information in the queue and send only the latest state.

Starting from Release 7.9.1, the previous implementation of slow peer is not supported.

Restrictions

Slow peer detection and processing is not intended for peers that are permanently slow. Peers that are permanently slow should be moved into their own update group by configuring same route-policy for all the permanent slow peers.

Types of Slow Peer

A slow peer can be configured as a static slow peer or a dynamic slow peer.

- When a peer is configured as a static slow peer, it is moved to its own update group isolating it from the other neighbors and hence static slow peers do not require any additional slow peer handling.
- When a peer is configured as a dynamic slow peer, it is grouped with other neighbors and hence requires slow peer detection and processing described in the [Dynamic Slow Peer](#) section.

Use of Refresh Sub-Group for Dynamic Slow Peer Processing

When a peer in an BGP sub-group is slow, it cannot keep up with the rate at which update messages are generated over time, causing formatted messages to accumulate. The rest of the members of the group that are faster than the slow peer and have completed transmission of the formatted messages will not be able to send new messages, even though there may be newly modified BGP nets waiting to be advertised or withdrawn.

This feature enables you to detect a slow peer in a sub-group and create a refresh sub-group to process the updates of the slow peer up to the current table version. New updates are still processed as part of the parent sub-group for all peers (slow and non-slow peers). Once all the updates in the refresh sub-group are sent to the slow peer, the refresh sub-group is deleted.

When a slow peer is detected, it automatically creates a new refresh sub-group to process the current updates, thus freeing the old messages in the parent sub-group and allowing the parent sub-group to process new

messages, thereby allowing other peers that are not slow to advance. If multiple peers are slow, each slow peer will be in its own refresh sub-group. When the update is finished, the refresh sub-group is removed.

Slow Peer State

Slow peer can be in one of the following detection or operational states. This state is influenced by both slow peer configuration and slow-peer detection or processing logic.

- *Static Slow Peer*: Neighbor address-family is in static slow peer state
- *Dynamic Detected Slow Peer*: Neighbor address-family is in dynamic detected slow peer state
- *Not slow peer*: Neighbor address-family is in not a slow peer state



Note The **show bgp <address-family> update out neighbor <neighbor-address> detail** command displays the neighbor address-family configuration states listed above.

Slow Detection State: Dynamic Detected Slow Peer

Dynamic Slow Peer Identification Logic

The following conditions to be met to classify a dynamic peer as slow peer:

- There are at least some messages for which acknowledgements have not been received from neighbor.
- There are some pending messages yet to be written to TCP.
- The time since the last update message timestamp has exceeded the configured threshold (default is 300 seconds)
- The number of refresh sub-groups for the address-family has not exceeded 16.
- The neighbor address-family is not the only member of the sub-group.
- All the other members of the sub-group are already marked as slow peer.
- There are some nets for which updates are yet to be generated.

When the above conditions are met, a peer is scheduled for slow peer processing by creating a refresh sub-group.



Note The **show bgp neighbors <neighbor-address> detail** command displays the *processing* states of the neighbor address-family configuration.

Processing slow peer: { TRUE / FALSE } indicates the slow peer processing state of the neighbor address-family. If the neighbor address-family is processed as a slow peer, TRUE; otherwise, FALSE.

Once the *Processing slow peer* state is set to TRUE, it is cleared only when it has completed processing of slow peer updates. It is not cleared even when any slow peer configuration changes for that neighbor address-family.

Dynamic Slow Peer Recovery Logic

A peer recovers from being classified as a Slow Peer when all messages queued for slow peer processing have been advertised and acknowledged by the peer.

Detection and Queue Management

When a peer is detected as slow, all messages queued in the peer's main queue are moved to a separate parallel slow peer queue. These messages are advertised separately, while new messages continue to be queued and advertised from the peer's main queue.

If the peer is detected as slow again while already being processed as a Slow Peer, the process of moving messages to the parallel slow peer queue and separate advertisement is repeated. This process continues until the peer is no longer detected as slow.

During the simultaneous processing of messages in both the main queue and the parallel slow peer queue, the order of advertisement for updates and withdrawals are maintained for each route.

Recovery

Once the Slow Peer has completed processing (advertised and acknowledged by the peer) all the messages in the slow peer queue, the slow peer queue is deleted, and the peer is no longer considered a Slow Peer.



Note During update generation, when a route is churning, the main queue may contain multiple withdrawals and updates for the route. When messages are moved from the main queue to the slow peer queue, only the last state of the route is queued to the parallel slow peer queue. This approach is advantageous when routes are churning in the network.

Refresh Sub-Group State

The refresh sub-group is a child of a sub-group. Refresh sub-groups are created for one or more neighbors in a sub-group to process the following special events:

- Route refresh request
- RT constraint change request
- Neighbors identified as slow peers

Normally, the refresh sub-group handles updates up to the current table version (target version), and the parent sub-group handles any new updates. Once the refresh sub-group finishes processing updates, it is deleted, and the parent sub-group continues to process the updates for all neighbors in the sub-group.

Refresh sub-group can be in one of the following states:

- *Not-In-Refresh*: Refresh sub-group is not present
- *RR*: Refresh sub-group is processing refresh request update
- *SLOW*: Refresh sub-group is processing slow-peer update
- *RTC*: Refresh sub-group is processing RTC incremental update
- *SLOW-RTC*: Refresh sub-group is processing both slow peer and RTC incremental update



Note The **show bgp <address-family> update out neighbor <neighbor-address> detail** command displays the refresh sub-group states listed above.

Slow Peer Configurations

Slow peer configuration can be enabled by any of the following:

- BGP global slow peer configuration - this configuration takes effect on all BGP peer neighbors
- BGP neighbor address-family configuration - this configuration takes effect only on a specific BGP neighbor

Slow Peer (BGP Global Configuration)

Configuration

This example below shows how to enable dynamic slow peer on all (default VRF and non-default VRF) BGP neighbor address-families:

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#slow-peer dynamic
Router(config-bgp)#commit
```

This example below shows how to disable slow peer on all (default VRF and non-default VRF) BGP neighbor address-families:

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#slow-peer detection-disable
Router(config-bgp)#commit
```

This example below shows how to enable dynamic slow peer with detection threshold of 120 seconds on all (default VRF and non-default VRF) BGP neighbor address-families:

```
Router#configure
Router(config)#router bgp 100
Router(config-bgp)#slow-peer dynamic threshold 120
Router(config-bgp)#commit
```

Running Configuration

```
router bgp 100
  slow-peer dynamic
!
!

router bgp 100
  slow-peer detection-disable
!
!

router bgp 100
  slow-peer dynamic threshold 120
!
!
```


Verification

The **show bgp neighbors <neighbor-address> detail** command displays the effective BGP neighbor slow peer configuration.

Slow Peer (BGP Neighbor Address-Family Configuration)

Configuration

This example below shows how to configure static slow peer for a (default VRF and non-default VRF) BGP 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 static
Router(config-bgp-nbr-af)#commit
```

This example below shows how to disable slow peer for a (default VRF and non-default VRF) BGP 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
```

This example below shows how to enable dynamic slow peer for a (default VRF and non-default VRF) BGP 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
```

This example below shows how to enable dynamic slow peer with detection threshold of 120 seconds for a (default VRF and non-default VRF) BGP 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
```

Running Configuration

```
router bgp 100
  neighbor 50.0.0.1
    address-family ipv4 unicast
      slow-peer static
    !
  !
!
```

```
router bgp 100
  neighbor 50.0.0.1
```

```

address-family ipv4 unicast
  slow-peer dynamic disable
!
!
!

router bgp 100
  neighbor 50.0.0.1
    address-family ipv4 unicast
      slow-peer dynamic
    !
  !
!

router bgp 100
  neighbor 50.0.0.1
    address-family ipv4 unicast
      slow-peer dynamic threshold 120
    !
  !
!

```

Verification

The **show bgp neighbors <neighbor-address> detail** command displays the effective BGP neighbor address-family slow peer configuration.

Slow Peer Effective Configuration State

You can enable slow peer configuration either by using global router configuration mode or by using neighbor address-family (AF) configuration mode.

The following 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*.

		Global configuration		
		[None]	[Dynamic]	[Detection disable]
Neighbor address-family configuration	[None]	<i>Detection-only</i>	<i>Dynamic</i>	<i>None</i>
	[Static]	<i>Static</i>	<i>Static</i>	<i>Static</i>
	[Dynamic]	<i>Dynamic</i>	<i>Dynamic</i>	<i>Dynamic</i>
	[Dynamic Disable]	<i>Detection-only</i>	<i>None</i>	<i>None</i>

The effective neighbor address-family configuration state can be any of the following entries in the table

AF configuration states	Details
<i>Static</i>	When the effective neighbor address-family configuration is <i>Static</i> , then that neighbor address-family is moved into its own unique update-group, thus isolating this neighbor address-family from other neighbors. If the user's intention is to group all the slow-peers into a single update group, it can be accomplished by removing static slow peer configuration and configuring the same neighbor out route-policy for all the neighbors.
<i>Dynamic</i>	When the effective neighbor address-family configuration is <i>Dynamic</i> , that BGP neighbor address-family is enabled for dynamic slow peer processing. When a neighbor address-family is enabled for dynamic slow peer processing, and the neighbor address-family is detected as slow, the neighbor address-family is processed in its own refresh sub-group without affecting other neighbors in the sub-group, in addition to displaying an IOS message indicating the neighbor address-family has become slow.
<i>Detection-only</i>	When the effective neighbor address-family configuration is <i>Detection-only</i> , whenever the neighbor address-family is detected as slow or recovers from being slow, an IOS message is displayed, but there will not be any mitigation to handle the slow peer.
<i>None</i>	When the effective neighbor address-family configuration is <i>None</i> , slow peer handling is disabled for that BGP neighbor address-family.



Note The **show bgp neighbors <neighbor-address> detail** command displays the neighbor address-family configuration states listed above.

This example below shows how to configure static slow peer for a (default VRF or non-default VRF) BGP neighbor address-family, while enabling dynamic slow peer on all other (default VRF and non-default VRF) BGP neighbor address-families:

```
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
```

This example below shows how to disable slow peer for a (default VRF and non-default VRF) BGP neighbor address-family, while enabling dynamic slow peer on all other (default VRF and non-default VRF) BGP neighbor address-families:

```
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
```

This example below shows how to enable dynamic slow peer with detection threshold of 120 seconds for a (default VRF and non-default VRF) BGP neighbor address-family, while enabling dynamic slow peer with detection threshold of 600 seconds on all other (default VRF and non-default VRF) BGP neighbor address-families:

```
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

The system generates the following log message when a peer is detected as a slow peer:

```
BGP neighbor 50.0.0.1 of vrf default afi IPv4 Unicast is detected as
slow-peer
```

The system generates the following log message when a slow peer recovers:

```
Slow BGP peer 50.0.0.1 of vrf default afi IPv4 Unicast has recovered
```

BGP Slow Peer Automatic Isolation from Update Group

Table 51: Feature History Table

Feature Name	Release Information	Feature Description
BGP Slow Peer Automatic Isolation from Update Group	Release 7.3.1	<p>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.</p> <p>New commands introduced in this release:</p> <ul style="list-style-type: none"> • slow-peer detection enable • clear bgp slow-peers <p>Updated commands in this release:</p> <ul style="list-style-type: none"> • slow-peer detection disable

The BGP Slow Peer Automatic Isolation from Update Group feature enables you to detect a slow peer in an update group and moving it to its own update group.

When a peer is slow in an BGP update group it cannot keep up with the rate at which update messages are generated over a prolonged time causing formatted messages to build up. The rest of the members of the group that are faster than the slow peer and have completed transmission of the formatted messages will not have anything new to send even though there may be newly modified BGP nets waiting to be advertised or withdrawn.

This feature enables you to detect a slow peer in an update group and moves it to its own update group. This feature is enabled by default.

When a slow peer is detected it is automatically moved to a new update group. Hence if there are slow peers then there will be an update group containing one or more slow peers corresponding to the original update group. There will be only one update group containing slow peers corresponding to the original update group. Hence, if multiple peers are slow, they will be in different sub-groups within the new slow update group. On recovery of the slow peer the peer is moved back to the original update group.

The presence of a slow peer in an update group, increases the number of formatted updates that are pending transmission. Events causing large churn in the BGP table, such as connection resets can result in a short-lived spike in the rate of update generation. A peer that temporarily falls behind during such events but quickly recovers after the event is not considered a slow peer.

This feature enables moving all the slow peers out of their original group, and into a new group dedicated to slow peers. After the slow peers are moved out, the non-slow members in the original group progress at their regular pace and catch up with the BGP table changes. The slow members consume updates at the slower pace and lag in their new dedicated group. One group for slow peers is required for each original group containing a slow peer. It is not possible to group together slow peers from different original groups as they will have a different outbound policy configuration.

Both the feature and splitting of update groups is enabled by default.

Configuration Examples

Detect Dynamic Slow Peers at the Global Configuration Level

Perform the following steps to disable slow peer detection globally and override all configuration under the neighbor. Any slow peers that are detected are marked as normal peers. They are moved back to their original update groups. No more slow peers are detected.

```
Router# configure
Router(config)# slow-peer-detection disable
```

Manually Configure Static Slow Peers at the Neighbor Configuration Level

Perform the following steps to control the behavior of the slow-peer detection and mitigation at neighbor configuration level. The configuration manually marks a neighbor as slow peer. Also, the peer will be part of slow update group.

```
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
```

Configure Dynamic Slow Peers at the Neighbor Configuration Level

Use the split-update-group dynamic command to dynamically detect the slow peer and move it to a slow update group.



Note When the split-update-group dynamic command alone is configured, the dynamically detected slow peer is moved to a slow update group. If there already exists a slow peer update group, the dynamic slow peer is moved to slow peer update group, otherwise a new slow peer update group is created and the peer is moved to the new slow peer update group. This option is enabled by default.



Note If the permanent keyword is not configured, the slow peer is moved to its regular original update group, after it becomes regular peer. If the permanent keyword is configured, the peer will not be moved to its original update group automatically. The administrator can use clear command to move it to original update group. Use this option if a peer keeps becoming a slow peer and recovering.

```
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
```

Clear Dynamically Detected Slow Peers

Perform the following task to clear all slow peers part of a specific address family identifiers (AFI) or subsequent address family identifiers (SAFI):

```
Router# clear bgp slow-peers <afi> <safi>
```

Perform the following task to clear all slow peers for all AFI or SAFI of the neighbor:

```
Router# clear bgp slow-peers <neighbor-address>
```

Perform the following task to clear the specified combination:

```
Router# clear bgp slow-peers <afi> <safi> <neighbor-address>
```

Running Configuration

This section shows the BGP Slow Peer Automatic Isolation from Update Group running configuration.

```
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
```

Verification

```
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

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
```

Configuring an Aggregate Contributor

Table 52: Feature History Table

Feature Name	Release	Description
Configuring an Aggregate Contributor	Release 7.5.4	<p>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.</p> <p>Earlier, there was no mechanism to identify a more specific route contributing to an aggregate and mark them as aggregate contributors.</p> <p>This feature introduces these changes:</p> <ul style="list-style-type: none"> • Introduces the show bgp aggregate-contributors command • Modifies the route-policy command

Route aggregation in Border Gateway Protocol (BGP) allows you to combine several specific routes into one route. You can then 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 the BGP attributes (for example, cost community, next-hop, BGP multiple exit discriminator) to the aggregate contributor that checks the integrity of BGP updates in BGP update messages and optimizes reaction when detecting invalid attributes, and apply the inbound policy and the outbound policy to the neighbors.

Restrictions

This feature is applicable for the following Address Family Indicators (AFIs):

- IPv4 unicast
- IPv6 unicast

Configuration

For the purpose of configuration examples for this feature:

- Router1 is where you configure the related aggregate contributor functionality and set the aggregate contributor flag.
- Router2 receives the routes with the modified BGP attributes you configured in Router1.

Set an aggregate contributor to a route policy

1. Enter the IOS XR configuration mode.

```
Router1# configure
```

2. Enter the aggregate route policy.

```
route-policy aggregate-route-policy-name
```

```
Router1(config)#route-policy aggregate-policy1
```

3. Set an aggregate contributor to the policy.

```
Router1(config-rpl)#set aggregate-contributor
```

4. End the policy, and then commit the changes.

```
Router1(config-rpl)#end
```

```
Router1(config-rpl)#commit
```

Set an aggregate contributor to a specific aggregate address route

1. Enter the IOS XR configuration mode.

```
Router1# configure
```

2. Specify the BGP Autonomous System Number (AS Number).

```
Router1(config)# router bgp 100
```

3. Specify the IP addresses from the address family (IPv4 unicast, IPv6 unicast) options.

```
Router1(config-bgp)# address-family ipv4 unicast
```

4. Specify the aggregate address followed by the aggregate route policy name.

```
aggregate-address address/mask-length route-policy aggregate-route-policy-name
```

```
Router1(config-bgp-af)#aggregate-address 250.2.2.0/24 route-policy aggregate-policy1
```

5. Commit the changes.

```
Router1(config-bgp-af)#commit
```

Running Configuration

```
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
    !
    !
    !

```

Verification

The following example displays the prefix as an aggregate contributor after applying the aggregate routing policy:

```

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)

```

The following output displays all aggregate contributors of the BGP address 250.2.2.0/24:

```

Router1#show bgp 250.2.2.0/24 aggregate-contributors
Thu Dec 1 10:15:20.393 EST
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

Setting BGP Attributes to the Aggregate Contributor

After setting an aggregate contributor to a specific aggregate address, you can then set or modify the BGP attributes (for example, cost community, next-hop, BGP multiple exit discriminator) to the aggregate contributor, and apply the inbound policy and the outbound policy to the neighbors.

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 would not be set in the remote node by default.

This is an optional step.

Configuration Example

For inbound policy

```

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

```

For outbound policy

```

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-else)# 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

```

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
  !
!

```

Verification

For inbound policy

The following example shows the BGP attribute (community) is 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**.

```

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)

```

For outbound policy

The following example shows the BGP attribute (extended-community) is 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 (For example, *Router2*) receives the route.

```

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

```

Management Information Base (MIBs) for BGP

Cisco IOS XR supports full MIBs and traps for OSPFv2/v3, as defined in RFC 4273. The RFC 4273 defines objects of the Management Information Base (MIB) for use with the BGP Routing Protocol.

To know more about MIBS, please use the [MIB Locator](#).

Peering Between BGP Routers Within a Confederation

Table 53: Feature History Table

Feature Name	Release Name	Description
Peering Between BGP Routers Within the Same Confederation	Release 7.11.1	<p>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.</p> <p>The feature introduces these changes:</p> <p>CLI:</p> <p>New Command:</p> <ul style="list-style-type: none">• <code>allowconfedas-in</code> <p>YANG Data Models</p> <ul style="list-style-type: none">• New XPaths for <code>Cisco-IOS-XR-ipv4-bgp-cfg.yang</code>• <code>Cisco-IOS-XR-um-router-bgp-cfg</code> <p>(see GitHub, YANG Data Models Navigator)</p>

Overview

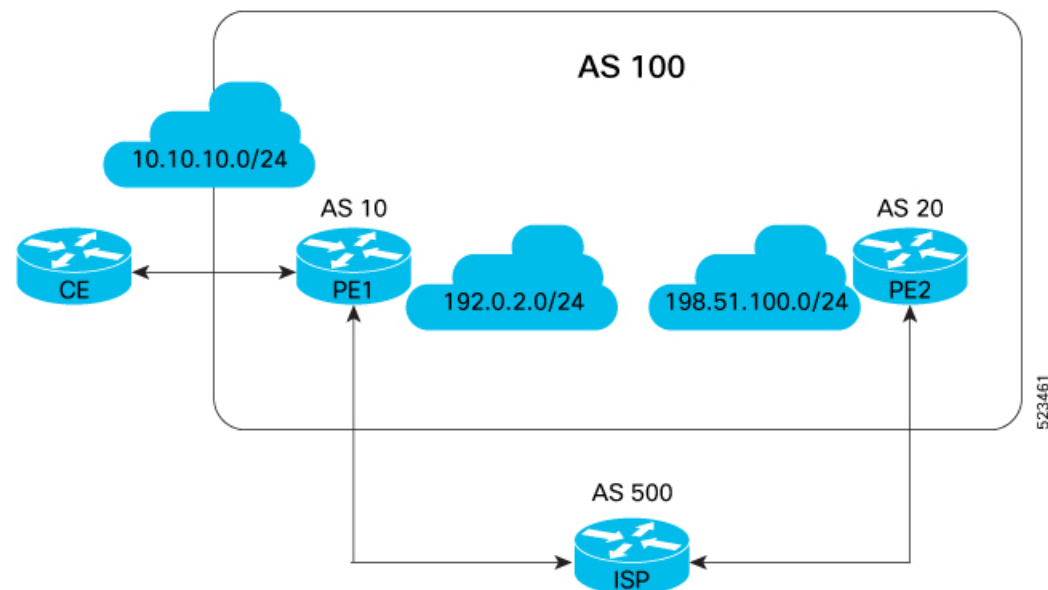
This feature, with its ability to enable BGP peering between routers in the sub-autonomous system (AS) within a confederation allows for specific router updates to be advertised using iBGP. This ensures that in the mesh of routers between sub-ASes in a confederation, the routers maintain consistent routing tables, and ensure proper reachability between networks within the confederation. To enable this feature, the users need to configure the **allowconfedas-in** command, thus circumventing the split horizon rule. You can specify the number of times the peer routers in the confederation can learn from each other when you configure the **allowconfedas-in** command.

In specific scenarios necessitating routing customization and optimization, breaking the split horizon rule is necessary. This rule restricts routers from sharing routes within the confederation. This feature allows you to

achieve that. You can configure the **allowconfedas-in** command to permit peers to learn routes from the same confederation.

In the topology illustrated in Figure 1: Peering Between BGP Routers Within the Same Confederation, the PE1 router connects to the ISP router via the 192.0.2.0/24 prefix, while the PE2 router connects via the 198.51.100.0/24 prefix. The CE router advertises the 10.10.10.0/24 route to PE1, which, in turn, advertises it to PE2. To achieve this, PE1 advertises the route to the ISP router, which then passes it to PE2 since PE1 and PE2 aren't directly connected. While relaying the advertisement, the ISP router learns the route. PE2, with a confederation AS number of AS 20, examines the AS number list in the advertisement to understand the route's path. PE2 identifies the AS numbers of the ISP router, which is AS 500, and PE1 router, which is AS 100. As the AS numbers of both PE1 and PE2 routers match, indicating they belong to the same confederation, PE2 drops the route in accordance with the split horizon rule. Hence these routers do not learn each others routes. The PE1 and PE2 routers are part of the same confederation and have different AS numbers. In this case, the **allows-in** command, which prevents dropping of the routes coming from a peer router of the same autonomous system, is not enough to allow the loop detection to be bypassed. Because of this, PE2 will not be able to learn prefixes from PE1 router. To override the split horizon rule and prevent PE2 from discarding the learned route, configure the **allowconfedas-in** command on both the PE1 and PE2 routers. The **allowconfedas-in** command enables you to configure the frequency with which peer routers within the same confederation learn from each other.

Figure 21: Peering Between BGP Routers Within the Same Confederation



Terminology

Autonomous Systems:

BGP, operating as an Exterior Gateway Protocol (EGP), establishes loop-free interdomain routing between autonomous systems (AS). An AS comprises routers under single administration, utilizing IGP for internal routing. Additionally, it employs EGP to route packets beyond its boundaries.

Sub-Autonomous System

A sub-autonomous system is a distinct subset within a larger autonomous system, possessing individual administrative control. It operates with specific routing policies, contributing to the hierarchical organization and efficient management of network configurations.

Confederation:

To reduce the iBGP mesh, an autonomous system can be segmented into sub-autonomous systems organized into a confederation. Externally, this confederation appears as a single autonomous system. Internally, each autonomous system is fully meshed but maintains limited connections to others in the same confederation. Peers in different autonomous systems engage in eBGP sessions, exchanging routing information resembling iBGP peers, preserving vital parameters like next hop, MED, and local preference.

Autonomous System Number

The Autonomous System Number (ASN) is crucial in networking, serving as a unique identifier for autonomous systems, including sub-autonomous systems within a confederation.

Split Horizon

Split horizon, a network protocol routing rule, boosts stability by prohibiting routers in the same confederation from sharing routes. It prevents a router from advertising routes back to the network from which it learned them. This prevents potential loops, ensuring accurate network topology views and enabling efficient data forwarding, thereby addressing routing issues.

Restrictions for Peering Between BGP Routers Within the Same Confederation

Peer routers within a confederation are restricted in the frequency at which they can exchange information with each other on configuring the **allowconfedas-in** command. The number of times they can share information ranges from 1 to 10. The default value is 3.

Configure Peering Between BGP Routers Within the Same Confederation

Configuration Example

To enable peering between routers that exist in the same confederation, perform the following steps:

- Enter router configuration mode.
- Assign BGP autonomous systems belonging to a confederation.
- Assign an identifier to the confederation.
- Place the router in neighbor configuration mode for routing and configure the neighbor IP address as a BGP peer.
- Specify either the IPv4 or IPv6 address family and enter address family configuration submode.
- Enable peer routers in the same confederation to learn from each other for a specified number of times.

```
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
```

Running Configuration

```
router bgp 65001
  bgp confederation peers 65002
```



```

bgp confederation identifier 100
neighbor 198.51.100.3
  address-family ipv4 unicast
    allowconfed-as-in 1

```

Verification

Verify the learning of routes among BGP peers. This output shows that the peers within the same confederation have learned from each others' routes, and the learning among peers has occurred thrice.

```

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.

```

Per-VRF label allocation for VPN routes

Table 54: Feature History Table

Feature Name	Release Information	Feature Description
Per-VRF label allocation for VPN routes	Release 24.4.1	<p>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])</p> <p>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.</p> <p>This feature introduces these changes:</p> <p>CLI:</p> <ul style="list-style-type: none"> advertise vpn-imported label-mode per-vrf <p>YANG Data Model:</p> <ul style="list-style-type: none"> Cisco-IOS-XR-um-vrf-cfg.yang <p>(see GitHub, YANG Data Models Navigator)</p>

Default label allocation behavior

When a Route Reflector (RR) that also serves as a Provider Edge (PE) router is configured with the same route distinguisher (RD) and the next-hop-self option, the system defaults to using per-prefix mode for local label allocation. This default behavior applies regardless of whether Prefix Independent Convergence (PIC) is enabled. In such cases, the local label allocation also defaults to per-prefix mode for remote VPN routes that share the same RD and have matching Route-Targets.

Label exhaustion in low-end platforms

On certain low-end platforms with limited label capacity, using per-prefix label allocation mode for imported VPN routes can cause label exhaustion. To mitigate this risk, you can use the Per-VRF Label Allocation for VPN Routes feature, which helps conserve label space.

Modified label allocation behavior

This feature modifies the label allocation behavior for imported VPN routes. If you intend to use per-VRF label allocation for imported VPN routes with the same RD, you can use the **advertise vpn-imported label-mode per-vrf** command. This command overrides the default per-prefix label allocation, enforcing per-VRF label allocation for your imported VPN routes.

Support for Per-VRF label allocation for VPN routes with different RDs

In scenarios involving different RDs, the default behavior assigns per-prefix local labels to routes with remote RDs, and these routes are advertised. However, imported VPN routes are not advertised by default.

When this feature is enabled, routes with remote RDs do not receive labels and are not advertised, whereas imported VPN routes are assigned a per-VRF label and are advertised.

Limitations for Per-VRF label allocation for VPN routes

These limitations apply to the Per-VRF label allocation for VPN routes feature:

- Applicable only to VPNv4 and VPNv6 routes.
- Applicable only to VRF-imported prefixes.
- Applicable only if 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.
- Not applicable to EVPN.
- The **is-imported-path** keyword for import match is only permitted at the neighbor outbound route-policy attach-point.

Configure Per-VRF label allocation for VPN routes

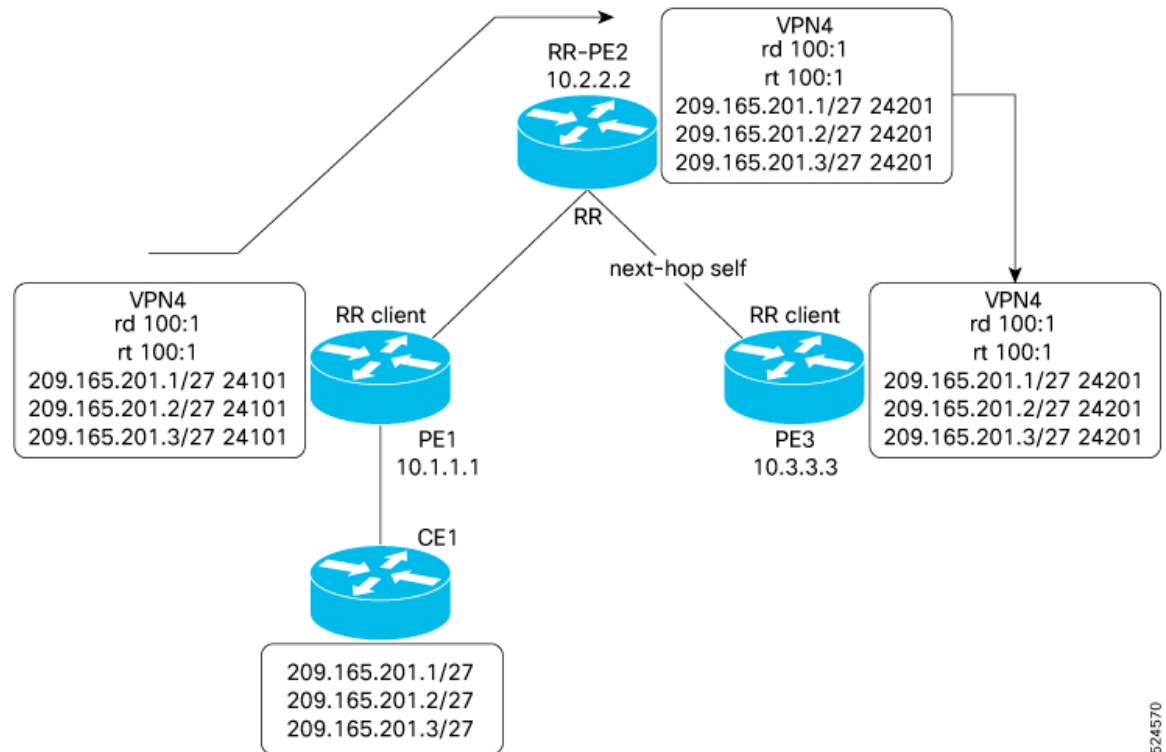
You can configure the Per-VRF label allocation for VPN routes feature in two scenarios:

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

Configure Per-VRF label allocation for VPN routes - same RD scenario

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

Figure 22: Sample topology for same RD configuration



524570

When using the same RD for RR and different provider edge routers, follow these configuration steps:

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 in this example, or you can set it within a neighbor outbound route-policy.

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

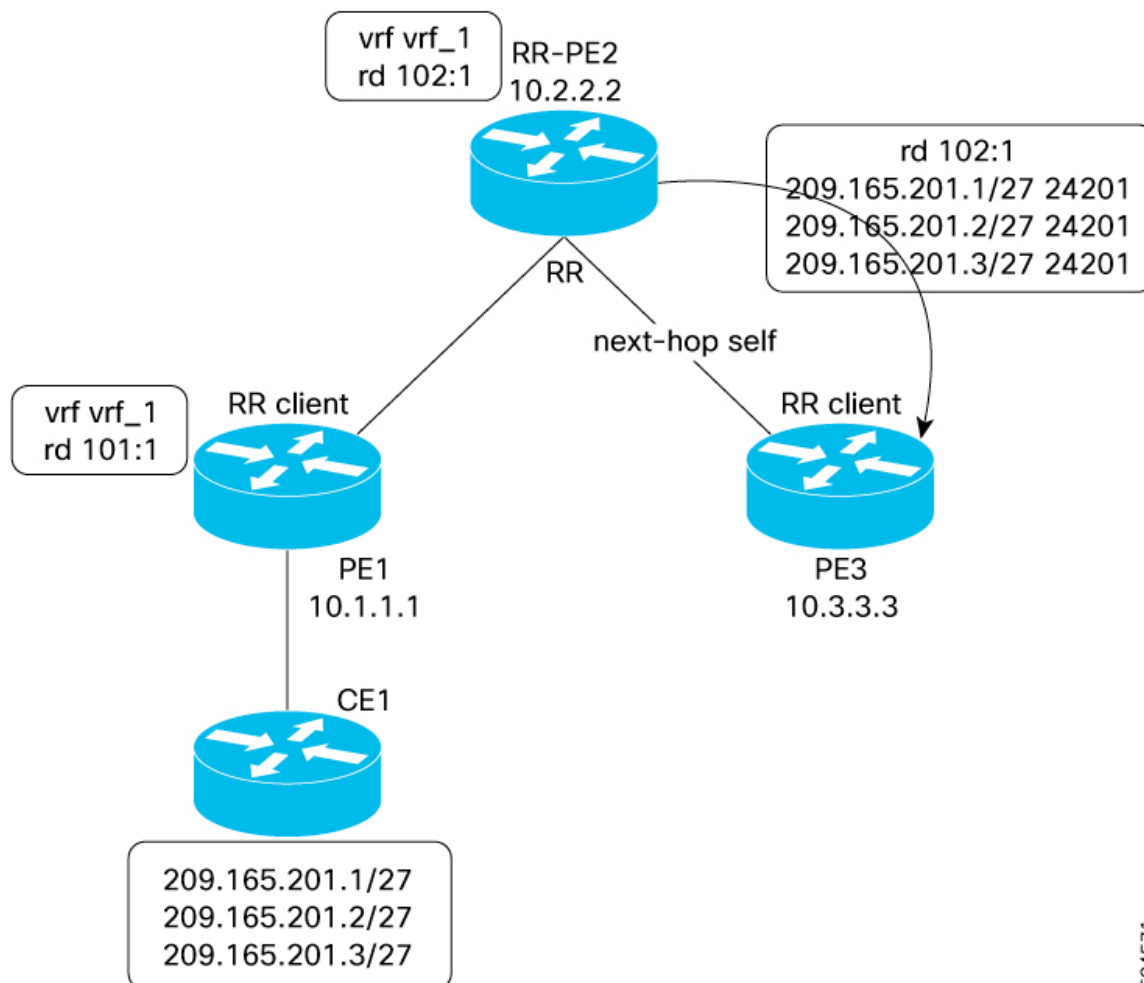
Example:

```
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 - different RD scenario

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

Figure 23: Sample topology for different RD configuration



When using different RDs for RR and provider edge routers, follow these steps:

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
```

Step 3 Create and apply route policies that specifically allow only imported VPN paths and local paths (including redistributed paths and Customer Edge (CE) routes) while 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
```

Virtual Routing Forwarding Next Hop Routing Policy

Table 55: Feature History Table

Feature Name	Release Name	Description
Virtual Routing Forwarding Next Hop Routing Policy	Release 7.11.1	<p>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.</p> <p>The feature introduces these changes:</p> <p>CLI:</p> <p>Modified Command:</p> <ul style="list-style-type: none">• The <code>nexthop route-policy</code> command is extended to VRF address-family configuration mode. <p>YANG Data Model</p> <ul style="list-style-type: none">• New XPaths for <code>Cisco-IOS-XR-ipv4-bgp-cfg.yang</code>• <code>Cisco-IOS-XR-um-router-bgp-cfg</code> <p>(see GitHub, YANG Data Models Navigator)</p>

Configure VRF Next Hop Policy

To enable next hop route policy on a VRF table, perform the following steps:

- Configure a route policy and enter route-policy configuration mode.
- Define the route policy to help limit notifications delivered to BGP for specific prefixes.
- Drop the prefix of the routes that matches the conditions set in the route policy.
- Enable BGP routing and enter the router configuration mode.
- Configure a VRF.

- Configure an IPv4 or IPv6 address family.
- Configure route policy filtering using next hops.

```
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
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
```

Running Configuration

```
route-policy nh-route-policy
  if destination in (10.1.1.0/24) and protocol in (connected, static) then
    drop
  endif
end-policy
!

router bgp 500
  vrf vrf10
    address-family ipv4 unicast
      nexthop route-policy nh-route-policy
```

Verification

Verify that the configured next route hop policy is enabled in a VRF table. The "BGP table nexthop route policy" field indicates the route policy used to determine the next hop for BGP routes in the specified VRF instance VRF1.

```
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  ?
*> 198.50.100.0/24  10.1.1.1      0              101  i
```

Enhanced Monitoring of NSR Statistics

Table 56: Feature History Table

Feature Name	Release Name	Description
Enhanced Monitoring of NSR Statistics	Release 24.2.11	Introduced in this release on: Fixed Systems (8700) (select variants only*) *Enhanced Monitoring of NSR Statistics is now supported on Cisco 8712-MOD-M routers.
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)

Overview

Non-Stop Routing (NSR) messages refer to the communication packets exchanged between networking devices to support Non-Stop Routing functionality. Non-Stop Routing is a feature designed to ensure uninterrupted network operation during software upgrades or failover events. NSR facilitates uninterrupted traffic forwarding by synchronizing state information between the primary and standby routing engines or

control planes within the device. This synchronization process ensures both components maintain consistent routing and forwarding data, providing seamless failover and reducing disruption to network traffic.

The Enhanced Monitoring of NSR Statistics feature offers network administrators a detailed snapshot of packet processing metrics, especially during critical operations like software upgrades or failover events. It provides comprehensive data on Non-Stop Routing (NSR) activities, becoming an essential tool for maintaining BGP performance and health with NSR enabled. This feature enables monitoring of BGP update processing speed, traffic volume, and packet sequence number integrity. Such insights are vital for ensuring network stability by detecting packet loss or sequencing inconsistencies. Early identification of processing delays or sequence number irregularities serves as a preemptive alert system, prompting necessary interventions to uphold BGP routing reliability and efficiency in the network infrastructure.

Insights During Critical Operations

During pivotal network events like software upgrades or failovers, precise real-time monitoring is essential. This feature allows administrators to closely observe and evaluate the impact of these operations on packet flow and processing.

Comprehensive NSR Data Analysis

Non-Stop Routing (NSR) ensures continuous packet forwarding amid network changes or failures. This monitoring tool offers exhaustive NSR activity data, crucial for analyzing BGP protocol health and performance under NSR.

Monitoring Key Routing Metrics

Tracking BGP update processing speeds, traffic volume, and packet sequencing integrity provides essential metrics. These metrics are vital for maintaining smooth BGP operation, fundamental for internet traffic routing.

Stability and Reliability Assurance

Insights from the Enhanced Monitoring feature maintain network stability by identifying packet loss or sequencing discrepancies. Proactive measures rectify issues before escalation, safeguarding network stability.

Proactive Problem Detection

Acting as an early detection system, the tool identifies processing delays or sequencing anomalies, allowing swift preemptive action to address potential issues.

Maintaining Routing Integrity

Timely interventions uphold the reliability and efficiency of BGP routing, critical for network operation health, ensuring consistent, uninterrupted data flows.

View Enhanced Monitoring of NSR Statistics

Run the **show bgp nsr** command to obtain crucial metrics for assessing the efficiency and robustness of BGP operations with NSR functionality. This command allows you to monitor the speed of BGP update processing, quantify packet volume over defined intervals, and validate sequence number integrity. These metrics help prevent packet loss and maintain network stability.

```
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: IGP 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

This table describes the significant fields shown in the display.

Table 57: Fields pertaining to Enhanced Monitoring of NSR Statistics in the output of show bgp neighbor details command

Field	Description
NPL Packet Processing Statistics	Statistics that are specifically detailed are related to network packet processing.
Interval (sec)	The length of the time interval, in seconds, during which the statistics were recorded.
End-Time (sec)	The time at which the data collection interval ended, typically shown as a timestamp.
Avg Proc Num of time (μs)	The average time taken to process each packet during the interval, measured in microseconds (μs).
Number of Pkts	The total number of packets processed during the interval.
seq num [start - end]	The list of sequence numbers of the first and last packets processed during the interval, allowing for the verification of continuous packet processing and the identification of any potential packet loss or misordering within the interval.

Enhanced BGP Monitoring and Prefix Analysis

Table 58: Feature History Table

Feature Name	Release Name	Description
Store and analyze changes in the prefixes received from BGP peer	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700) (select variants only*)</p> <p>*The functionality of storing and analyzing changes in the prefixes received from BGP peer is now supported on Cisco 8712-MOD-M routers.</p>
Store and analyze changes in the prefixes received from BGP peer	Release 24.2.1	<p>You can now actively monitor changes, acceptances, and rejections of received prefixes by providing insightful statistics using this feature. It enables storing all original copies of routes received from peers, even those not chosen as the best path. Use the command to accomplish this, while the inbound RPL concurrently modifies received prefixes.</p> <p>The feature introduces these changes:</p> <p>CLI:</p> <ul style="list-style-type: none">• The soft-reconfig-stats keyword is added to the show bgp summary command.• The dryrun-policy keyword is added to the show bgp neighbor command. <p>YANG Data Model:</p> <ul style="list-style-type: none">• Cisco-IOS-XR-ipv4-bgp-oper (see GitHub, YANG Data Models Navigator)

Overview

For routers in certain customer deployments, robust support is needed for accommodating millions of BGP paths, encompassing both IPv4 and IPv6 unicast address-family identifiers and sub-address-family identifiers.

While the Cisco IOS XR BGP implementation can operate at this scale, there's a necessity for serviceability enhancements. These enhancements are vital for effectively monitoring BGP operations and facilitating debugging procedures, ensuring smooth deployment in both production environments and lab testing scenarios.

Monitor software reconfiguration and BGP dry run statistics

You can now actively monitor changes, acceptances, and rejections of received prefixes by providing insightful statistics using this feature. It enables storing all original copies of routes received from peers, even those not chosen as the best path.

Soft reconfiguration enables the software to store the incoming prefixes before you apply the policy if route refresh is not supported by the peer (otherwise a copy of the update is not stored). The **always** keyword forces the software to store a copy even when route refresh is supported by the peer.

Using the **soft reconfiguration inbound always** command, you can enable the software to store updates received from a specified neighbor. The **soft reconfiguration inbound** command causes the software to store the original unmodified route in addition to a route that is modified or filtered. This allows a “soft clear” to be performed after the inbound policy is changed.

The incoming prefixes are categorized into one of three types:

- Accepted and unmodified
- Accepted and modified
- Denied

As part of serviceability enhancements, this feature enables active monitoring of changes, acceptances, and rejections of received prefixes, offering detailed statistics. It retains original copies of all routes received from peers, even those not chosen as the best path.

Verification

Verify the soft reconfiguration statistics for IPv4 unicast BGP sessions.

```
Router# show bgp ipv4 unicast summary soft-reconfig-stats
```

```
Tue Sep 19 14:02:39.106 EDT
BGP router identifier 192.168.0.3, local AS number 3
BGP generic scan interval 60 secs
Non-stop routing is enabled BGP table state: Active
Table ID: 0x0000000 RD version: 6
BGP table nexthop route policy:
BGP main routing table version 6
BGP NSR Initial initsync version 6 (Reached)
BGP NSR/ISSU Sync-Group versions 6/0
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.

Process
Speaker    RcvTblVer  bRIB/RIB  Labelver  ImportVer  SendTblVer  StandbyVer

Neighbor    Spk   AS   MsgRcvd  MsgSent   TblVer   InQ    OutQ    Up/Down    St/PfxRcd  SoftChgd
Denied
10.10.10.4   0     3    15       12        6       0      0      00:46:06    2          0
0
Total                               2          0
```

0

Legend:

Total PfxRcd: Sum of accepted unmodified and modified paths

Total SoftChgd: Sum of accepted modified paths

Total Denied: Sum of Denied paths

Verify differential statistics between the existing inbound policy and the dry run policy to evaluate the impact on the prefix scale.

```
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. */
```

Verify the statistics on configured and established neighbors, and address-family prefixes, paths, and memory usage,

```
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 Memory	Path Memory	PathElem 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
```

Monitor BGP Memory Statistics

The Monitor BGP Memory Statistics functionality incorporates a periodic memory state check by the BGP process, with a timer set at 60-second intervals. It also logs memory changes in both the BGP trace and a circular buffer, facilitating the generation of easily understandable show outputs. This feature detects and records any change in memory utilization exceeding the 1% threshold of memory utilization (rlimit) since the last report. As the memory level approaches the rlimit and falls below it, gradual syslog notifications are triggered at thresholds of 85%, 90%, and 95% of the rlimit.

When you use the **show bgp memory history** command, you can learn from the output details of memory usage, percentage of rlimit utilized, memory variations between records, and calculations for networks, paths, path elements, and attributes, specifically for the default VRF.

```
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						

```
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

```

Enhanced Monitoring of BGP Keepalive Messages

Table 59: Feature History Table

Feature Name	Release Name	Description
Enhanced Monitoring of BGP Keepalive Messages	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700) (select variants only*)</p> <p>*Enhanced Monitoring of BGP Keepalive Messages is now supported on Cisco 8712-MOD-M routers.</p>

Feature Name	Release Name	Description
Enhanced Monitoring of BGP Keepalive Messages	Release 24.2.11	<p>You can now achieve enhanced stability and fairness in your network's traffic processing by utilizing Enhanced Monitoring of BGP Keepalive Messages feature. This functionality enables you to efficiently manage the influx of messages from multiple neighbors by meticulously monitoring and adjusting the rate of message processing through throttling. Take advantage of a comprehensive log detailing each neighbor's throttling incidents, empowering network administrators with critical data to refine and optimize overall network performance.</p> <p>CLI:</p> <p>The feature modifies the output of the show command given below:</p> <ul style="list-style-type: none"> • show bgp neighbor detail <p>YANG Data Model:</p> <ul style="list-style-type: none"> • Cisco-IOS-XR-ipv4-bgp-oper <p>(see GitHub, YANG Data Models Navigator)</p>

Overview

The Border Gateway Protocol (BGP) provides a feature that controls the flow of incoming messages from various neighbors, promoting stability and fairness in network traffic processing. It achieves this by monitoring and regulating the rate at which it processes messages—a process known as throttling. The feature records a detailed history of each neighbor's throttling events, giving network administrators valuable insights to optimize network performance.

Per-Neighbor Input Queue Management

Each BGP neighbor has its own input queue for storing packets received from that neighbor. The Input/Output Read thread populates this queue with incoming messages. If the packet volume exceeds a predefined threshold, BGP triggers throttling to reduce the message processing rate. This approach ensures fair processing time for each neighbor and prevents any single neighbor from overwhelming the system.

For each BGP neighbor, the system establishes a dedicated write queue to store outbound packets. The Input/Output thread is populates these packets into the corresponding neighbor's socket, contingent upon the availability of space within the TCP buffer. When the TCP buffer reaches its maximum capacity, BGP activates a throttling mechanism to temporarily halt the dispatch of messages. The suspension of message transmission ends upon the neighbor's acknowledgment of received packets and subsequent freeing of space in the TCP

buffer. At this point, a callback initiates, prompting BGP to resume the data writing operation to the socket. This process meticulously monitors and logs the throttled condition of each neighbor, ensuring an equitable distribution of network resources and maintaining transmission stability.

Throttled State Tracking and Reporting

This feature logs the precise times when a neighbor's input and output queues enters and exits the throttled state.

Throttling Peak Duration

This feature identifies the maximum duration for which a neighbor's input and output queue was throttled and records the time this occurred.

Historical Data Retention

A comprehensive historical record is maintained to aid in the analysis of throttling behavior over time:

Circular Buffer for Throttling Events: The last 10 throttling states for each neighbor are stored in a circular buffer. This includes the data outlined above for each event. The buffer is designed to persist even after the neighbor experiences a reset, providing a continuous record across sessions.

- **Throttling Duration Record:** Beyond the circular buffer, the longest recorded throttling duration is retained after a system reset. This creates a historical record of peak throttling events for each neighbor.

Keepalive Interval and Hold Timer

The keepalive interval specifies how often BGP sends keepalive messages to its neighbors. These messages confirm that the BGP session is still active and that the TCP connection is healthy. The typical default value for the keepalive interval is 60 seconds, but this can vary based on configuration.

Keepalive interval determines how often BGP sends "heartbeat" messages, and the hold timer determines how long BGP will wait for these messages before declaring the connection lost. The relationship between these two settings is key to maintaining stable BGP operations and minimizing disruptions due to lost connections

Hold time defines the maximum amount of time a BGP router wait for a keepalive message from its neighbor before considering the connection lost. It is a safety mechanism to detect and react to communication failures. The hold timer is usually set to a higher value, often three times the keepalive interval. If BGP does not receive a keepalive within the hold timer duration, it assumes the session has failed and takes corrective action.

This feature monitors parameters, such as:

- The maximum amount of time that has passed since the last BGP keepalive message was received from a neighbor before a BGP session failed.
- The 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.
- Timestamp when this maximum hold time was observed.
- The number of times the hold time crossed certain thresholds suggesting that the hold time reached a significant portion of its configured value but did not exceed it by a large margin, or

.When the keep alive message have been delayed from certain neighbours, the feature tracks how many times the hold time crossed certain thresholds, without receiving the keepalive message. For example, when 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. It also indicates that 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 that the hold time frequently approached significant portions of its configured maximum value before the BGP session reset.

View Enhanced Monitoring of BGP Keepalive Messages

Run the `show bgp neighbor detail` to learn detailed information of BGP session stability and message handling, detailing various critical timers and queue metrics, such as the maximum time intervals before a BGP session is considered down or reset due to lack of keepalive messages, initial message reception timestamps from neighbors, periods of read throttling, and statistics on output queue sizes and message processing during these throttle intervals.

Router# show bgp neighbor detail

Mon Aug 28 21:43:35.363 IST

```
BGP neighbor is 10.10.10.1
Remote AS 200, local AS 100, external link
Remote router ID 10.10.10.1
BGP state = Established, up for 00:30:44
NSR State: NSR Ready
Last read 00:00:02, Last read before reset 00:00:00
Hold time is 15, keepalive interval is 5 seconds
Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
Last write 00:00:03, attempted 19, written 19
Second last write 00:00:08, 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 Aug 28 21:43:33.440 last full not set pulse count 929
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 00:19:18, second last 00:19:18
Last KA expiry before reset 00:19:20, second last 00:00:0
Last KA error before reset 00:00:00, KA not sent 00:00:00
Last KA start before reset 00:19:18, second last 00:19:18

/* The highlighted output provide information about
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

```

Precedence: internet
Non-stop routing is enabled
Entered Neighbor NSR TCP mode:
  TCP Initial Sync :          Aug 28 21:13:59.314
  TCP Initial Sync Phase Two : Aug 28 21:14:00.321
  TCP Initial Sync Done :      Aug 28 21:14:01.326
Graceful restart is enabled
Restart time is 200 seconds
Stale path timeout time is 300 seconds
Enforcing first AS is enabled
Multi-protocol capability received
Neighbor capabilities:
  Route refresh:      Yes      Rcvd
  4-byte AS:         Yes      Yes
  Address family IPv4 Unicast: Yes      Yes
Message stats:
  InQ depth: 0, OutQ depth: 0
    Last_Sent      Sent  Last_Rcvd      Rcvd
Open:      Aug 28 21:12:49.241      1  Aug 28 21:12:51.244      1
Notification: ---      0  ---      0
Update:      Aug 28 21:12:57.318      5  Aug 28 21:12:57.326      2
Keepalive:   Aug 28 21:43:32.386     369  Aug 28 21:43:33.440     615
Route_Refresh: ---      0  ---      0
Total:      375      618
Minimum time between advertisement runs is 30 secs
Inbound message logging enabled, 3 messages buffered
Outbound message logging enabled, 3 messages buffered
Event specific version bump counts:
  Interval definitions:
    Interval  Duration (min)  Start time      End time
Live
1      5  Aug 28 21:37:44.370  Aug 28 21:42:44.370
2      15 Aug 28 21:27:44.370  Aug 28 21:42:44.370
3      30 Aug 28 21:12:44.370  Aug 28 21:42:44.370
Category      Total      Live  Interval 1  Interval 2  Interval
3
  Reachable      5      0      0      0
5
  Unreachable    0      0      0      0
0

For Address Family: IPv4 Unicast
BGP neighbor version 98670000062
Update group: 0.2 Filter-group: 0.2 No Refresh request being processed
AF-dependent capabilities:
  Graceful Restart capability advertised
    Local restart time is 200, RIB purge time is 600 seconds
    Maximum stalepath time is 300 seconds
  Extended Nexthop Encoding: advertised and received
Route refresh request: received 0, sent 0
Policy for incoming advertisements is pass
Policy for outgoing advertisements is pass
17 accepted prefixes, 5 are bestpaths
Accepted prefixes (modified and unmodified) high water mark : 17 (Aug 28 21:12:57.326)
Accepted prefixes modified : 0.
Accepted prefixes modified high water mark : 0 (not set)
Exact no. of prefixes denied : 0.
Denied prefixes high water mark : 0 (not set)
Cumulative no. of prefixes denied: 0.
Prefix advertised 56, suppressed 0, withdrawn 0
An EoR was received during read-only mode
Last ack version 98670000062, Last synced ack version 98670000062
Outstanding version objects: current 0, max 1, refresh 0

```

```

Additional-paths operation: None
Advertise routes with local-label via Unicast SAFI
Slow peer flags: 18
Event specific version bump counts:
  Category          Total          Live  Interval 1  Interval 2  Interval
3
  Reachable          5            0            0            0
5
  Unreachable         0            0            0            0
0

```

```

Connections established 1; dropped 0
Local host: 10.10.10.2, Local port: 38803, IF Handle: 0x01004020
Foreign host: 10.10.10.1, Foreign port: 179
Last reset 00:00:00
Address-Family Ref Count:
  IPv4 Unicast: 17

```

```

BGP neighbor is 192.168.0.5
Remote AS 100, local AS 100, internal link
Remote router ID 192.168.0.5
BGP state = Established, up for 00:30:45
NSR State: NSR Ready
Last read 00:00:03, Last read before reset 00:00:00
Hold time is 15, keepalive interval is 5 seconds
Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
Last write 00:00:03, attempted 19, written 19
Second last write 00:00:08, 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 Aug 28 21:43:32.665 last full not set pulse count 752
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
Entered Neighbor NSR TCP mode:
  TCP Initial Sync :      Aug 28 21:13:59.314
  TCP Initial Sync Phase Two :  Aug 28 21:14:00.321
  TCP Initial Sync Done :      Aug 28 21:14:01.326
Graceful restart is enabled
Restart time is 200 seconds
Stale path timeout time is 300 seconds
Multi-protocol capability received
Neighbor capabilities:
  Adv          Rcvd
Route refresh: Yes          Yes
4-byte AS:     Yes          Yes
Address family IPv4 Unicast: Yes          Yes
Address family VPNv4 Unicast: Yes          Yes
Address family IPv6 Labeled-unicast: Yes          Yes
Address family VPNv6 Unicast: Yes          Yes
Address family RT Constraint: Yes          Yes
Message stats:
  InQ depth: 0, OutQ depth: 0
  Last_Sent          Sent  Last_Rcvd          Rcvd
Open:               Aug 28 21:12:48.188          1  Aug 28 21:12:50.192          1
Notification:      ---          0  ---          0
Update:            Aug 28 21:12:57.364          30  Aug 28 21:12:57.343          76
Keepalive:          Aug 28 21:43:32.463          370  Aug 28 21:43:32.417          369
Route_Refresh:      ---          0  ---          0
Total:              401          446

```

```

Minimum time between advertisement runs is 0 secs
Inbound message logging enabled, 3 messages buffered
Outbound message logging enabled, 3 messages buffered
Event specific version bump counts:
  Interval definitions:
    Interval  Duration (min)  Start time          End time
  Live
    1           5 Aug 28 21:42:44.370 Aug 28 21:43:35.670
    2          15 Aug 28 21:27:44.370 Aug 28 21:42:44.370
    3          30 Aug 28 21:12:44.370 Aug 28 21:42:44.370
  Category              Total      Live  Interval 1  Interval 2  Interval
3
  Reachable              282         0         0         0
282
  Unreachable            0         0         0         0
0

```

```

For Address Family: IPv4 Unicast
BGP neighbor version 98670000062
Update group: 0.3 Filter-group: 0.1 No Refresh request being processed
AF-dependent capabilities:
  Graceful Restart capability advertised
    Local restart time is 200, RIB purge time is 600 seconds
    Maximum stalepath time is 300 seconds
  Extended Nexthop Encoding: advertised and received
Route refresh request: received 0, sent 0
33 accepted prefixes, 31 are bestpaths
Accepted prefixes (modified and unmodified) high water mark : 33 (Aug 28 21:12:55.231)
Accepted prefixes modified : 0.
Accepted prefixes modified high water mark : 0 (not set)
Exact no. of prefixes denied : 0.
Denied prefixes high water mark : 0 (not set)
Cumulative no. of prefixes denied: 0.
Prefix advertised 30, suppressed 0, withdrawn 0
AIGP is enabled
An EoR was received during read-only mode
Last ack version 98670000062, Last synced ack version 98670000062
Outstanding version objects: current 0, max 2, refresh 0
Additional-paths operation: None
Send Multicast Attributes
Advertise routes with local-label via Unicast SAFI
Slow peer flags: 18
Event specific version bump counts:
  Category              Total      Live  Interval 1  Interval 2  Interval
3
  Reachable              31         0         0         0
31
  Unreachable            0         0         0         0
0

```

```

For Address Family: VPNv4 Unicast
BGP neighbor version 98670000310
Update group: 0.2 Filter-group: 0.1 No Refresh request being processed
AF-dependent capabilities:
  Graceful Restart capability advertised
    Local restart time is 200, RIB purge time is 600 seconds
    Maximum stalepath time is 300 seconds
  Extended Nexthop Encoding: advertised and received
Route refresh request: received 0, sent 0
110 accepted prefixes, 110 are bestpaths
Accepted prefixes (modified and unmodified) high water mark : 110 (Aug 28 21:12:57.335)
Accepted prefixes modified : 0.
Accepted prefixes modified high water mark : 0 (not set)
Exact no. of prefixes denied : 0.

```



```

Denied prefixes high water mark : 0 (not set)
Cumulative no. of prefixes denied: 0.
Prefix advertised 55, suppressed 0, withdrawn 0
AIGP is enabled
An EoR was received during read-only mode
Last ack version 98670000310, Last synced ack version 98670000310
Outstanding version objects: current 0, max 1, refresh 0
Additional-paths operation: None
Send Multicast Attributes
Slow peer flags: 18
Event specific version bump counts:
  Category                Total      Live  Interval 1  Interval 2  Interval
3
  Reachable                110        0         0         0         0
110
  Unreachable              0         0         0         0         0
0

```

```

For Address Family: IPv6 Labeled-unicast
BGP neighbor version 98670000062
Update group: 0.2 Filter-group: 0.2 No Refresh request being processed
AF-dependent capabilities:
  Graceful Restart capability advertised
    Local restart time is 200, RIB purge time is 600 seconds
    Maximum stalepath time is 300 seconds
Route refresh request: received 0, sent 0
33 accepted prefixes, 31 are bestpaths
Accepted prefixes (modified and unmodified) high water mark : 33 (Aug 28 21:12:55.230)
Accepted prefixes modified : 0.
Accepted prefixes modified high water mark : 0 (not set)
Exact no. of prefixes denied : 0.
Denied prefixes high water mark : 0 (not set)
Cumulative no. of prefixes denied: 0.
Prefix advertised 30, suppressed 0, withdrawn 0
AIGP is enabled
An EoR was received during read-only mode
Last ack version 98670000062, Last synced ack version 98670000062
Outstanding version objects: current 0, max 1, refresh 0
Additional-paths operation: None
Send Multicast Attributes
Slow peer flags: 18
Event specific version bump counts:
  Category                Total      Live  Interval 1  Interval 2  Interval
3
  Reachable                31        0         0         0         0
31
  Unreachable              0         0         0         0         0
0

```

```

For Address Family: VPNv6 Unicast
BGP neighbor version 98670000310
Update group: 0.2 Filter-group: 0.1 No Refresh request being processed
AF-dependent capabilities:
  Graceful Restart capability advertised
    Local restart time is 200, RIB purge time is 600 seconds
    Maximum stalepath time is 300 seconds
Route refresh request: received 0, sent 0
110 accepted prefixes, 110 are bestpaths
Accepted prefixes (modified and unmodified) high water mark : 110 (Aug 28 21:12:57.343)
Accepted prefixes modified : 0.
Accepted prefixes modified high water mark : 0 (not set)
Exact no. of prefixes denied : 0.
Denied prefixes high water mark : 0 (not set)
Cumulative no. of prefixes denied: 0.

```

```

Prefix advertised 55, suppressed 0, withdrawn 0
AIGP is enabled
An EoR was received during read-only mode
Last ack version 98670000310, Last synced ack version 98670000310
Outstanding version objects: current 0, max 1, refresh 0
Additional-paths operation: None
Send Multicast Attributes
Slow peer flags: 18
Event specific version bump counts:
  Category          Total          Live  Interval 1  Interval 2  Interval
3
  Reachable          110           0           0           0
110
  Unreachable         0           0           0           0
0

For Address Family: RT Constraint
BGP neighbor version 98670000010
Update group: 0.2 Filter-group: 0.1 No Refresh request being processed RT constraint
nbr enabled for VPN updates:
AF-dependent capabilities:
  Graceful Restart capability advertised
    Local restart time is 200, RIB purge time is 600 seconds
    Maximum stalepath time is 300 seconds
Route refresh request: received 0, sent 0
10 accepted prefixes, 0 are bestpaths
Accepted prefixes (modified and unmodified) high water mark : 10 (Aug 28 21:12:55.219)
Accepted prefixes modified : 0.
Accepted prefixes modified high water mark : 0 (not set)
Exact no. of prefixes denied : 0.
Denied prefixes high water mark : 0 (not set)
Cumulative no. of prefixes denied: 0.
Prefix advertised 10, suppressed 0, withdrawn 0
AIGP is enabled
An EoR was received during read-only mode
Last ack version 98670000010, Last synced ack version 98670000010
Outstanding version objects: current 0, max 1, refresh 0
Additional-paths operation: None
Send Multicast Attributes
Slow peer flags: 18
Event specific version bump counts:
  Category          Total          Live  Interval 1  Interval 2  Interval
3
  Reachable           0           0           0           0
0
  Unreachable          0           0           0           0
0

Connections established 1; dropped 0
Local host: 192.168.0.2, Local port: 24212, IF Handle: 0x00000000
Foreign host: 192.168.0.5, Foreign port: 179
Last reset 00:00:00
Address-Family Ref Count:
  IPv4 Unicast: 33
  VPNv4 Unicast: 220
  IPv6 Unicast: 33
  VPNv6 Unicast: 220
  RT Constraint: 10

BGP neighbor is 10:10:10::1
Remote AS 200, local AS 100, external link
Remote router ID 10.10.10.1
BGP state = Established, up for 00:30:45
NSR State: NSR Ready

```

```

Last read 00:00:02, Last read before reset 00:00:00
Hold time is 15, keepalive interval is 5 seconds
Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
Last write 00:00:03, attempted 19, written 19
Second last write 00:00:08, 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 Aug 28 21:43:33.434 last full not set pulse count 949
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
Entered Neighbor NSR TCP mode:
    TCP Initial Sync :          Aug 28 21:13:59.314
    TCP Initial Sync Phase Two : Aug 28 21:14:00.322
    TCP Initial Sync Done :      Aug 28 21:14:01.326
Graceful restart is enabled
Restart time is 200 seconds
Stale path timeout time is 300 seconds
Enforcing first AS is enabled
Multi-protocol capability received
Neighbor capabilities:          Adv          Rcvd
Route refresh:                  Yes          Yes
4-byte AS:                      Yes          Yes
Address family IPv6 Unicast:    Yes          Yes
Message stats:
InQ depth: 0, OutQ depth: 0
      Last_Sent          Sent  Last_Rcvd          Rcvd
Open:      Aug 28 21:12:47.832      1  Aug 28 21:12:49.848      1
Notification:  ---          0  ---          0
Update:      Aug 28 21:12:57.326      4  Aug 28 21:12:57.329      2
Keepalive:   Aug 28 21:43:32.406     369  Aug 28 21:43:33.434     615
Route_Refresh:  ---          0  ---          0
Total:              374          618
Minimum time between advertisement runs is 30 secs
Inbound message logging enabled, 3 messages buffered
Outbound message logging enabled, 3 messages buffered
Event specific version bump counts:
Interval definitions:
Interval  Duration (min)  Start time          End time
Live              5  Aug 28 21:42:44.370  Aug 28 21:43:35.670
1                5  Aug 28 21:37:44.370  Aug 28 21:42:44.370
2               15  Aug 28 21:27:44.370  Aug 28 21:42:44.370
3               30  Aug 28 21:12:44.370  Aug 28 21:42:44.370
Category          Total      Live  Interval 1  Interval 2  Interval
3
Reachable              5          0          0          0
5
Unreachable            0          0          0          0
0

For Address Family: IPv6 Unicast
BGP neighbor version 98670000062
Update group: 0.3 Filter-group: 0.1 No Refresh request being processed
Community attribute sent to this neighbor
AF-dependent capabilities:
Graceful Restart capability advertised
Local restart time is 200, RIB purge time is 600 seconds
Maximum stalepath time is 300 seconds
Route refresh request: received 0, sent 0

```

```

Policy for incoming advertisements is pass
Policy for outgoing advertisements is pass
17 accepted prefixes, 5 are bestpaths
Accepted prefixes (modified and unmodified) high water mark : 17 (Aug 28 21:12:57.326)
Accepted prefixes modified : 0.
Accepted prefixes modified high water mark : 0 (not set)
Exact no. of prefixes denied : 0.
Denied prefixes high water mark : 0 (not set)
Cumulative no. of prefixes denied: 0.
Prefix advertised 56, suppressed 0, withdrawn 0
An EoR was received during read-only mode
Last ack version 98670000062, Last synced ack version 98670000062
Outstanding version objects: current 0, max 1, refresh 0
Additional-paths operation: None
Advertise routes with local-label via Unicast SAFI
Slow peer flags: 18
Event specific version bump counts:
  Category                Total          Live  Interval 1  Interval 2  Interval
3
  Reachable                5              0          0          0
5
  Unreachable              0              0          0          0
0

Connections established 1; dropped 0
Local host: 10:10:10::2, Local port: 28412, IF Handle: 0x01004020
Foreign host: 10:10:10::1, Foreign port: 179
Last reset 00:00:00
Address-Family Ref Count:
  IPv6 Unicast: 1

```

This table describes the significant fields shown in the display.

Table 60: 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.
Max Hold Time elapsed was 6001 msec at Sep 12 17:02:36.954, crossed 40%: 2, 70%: 0	<p>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.</p> <p>In this specific output, the fields indicate the following:</p> <p><i>Max Hold Time elapsed was 6001 msec:</i> indicates that the maximum time interval between receiving keepalive messages from the neighbor was 6001 milliseconds or approximately 6 seconds.</p> <p><i>at Sep 12 17:02:36.954:</i> Timestamp when this maximum hold time was observed.</p> <p><i>crossed 40%: 2, 70%: 0:</i> 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.</p>

Field	Description
Max Hold Time elapsed before reset was 9001 msec at Sep 12 17:01:53.397, crossed 40%: 7, 70%: 2	<p>Maximum duration between receiving BGP (Border Gateway Protocol) keepalive messages from a neighbor before the BGP session was reset.</p> <p>In this specific output, the fields indicate the following:</p> <p><i>Max Hold Time elapsed before reset was 9001 msec:</i> Maximum time interval between receiving keepalive messages from the neighbor before the BGP session reset was 9001 milliseconds or approximately 9 seconds.</p> <p><i>at Sep 12 17:01:53.397:</i> Timestamp when this maximum hold time before reset was observed.</p> <p><i>crossed 40%: 7, 70%: 2:</i> 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</p>
First message received at Sep 12 16:45:00.973, sent at Sep 12 16:45:00.975	<p>Timestamp when the first message from a BGP neighbor was received by the local router.</p> <p>In this specific output, the fields indicate the following:</p> <p><i>First message received at Sep 12 16:45:00.973:</i> First message from the BGP neighbor was received at 16:45:00 on September 12th</p> <p><i>sent at Sep 12 16:45:00.975:</i> Timestamp when the corresponding message was sent by the BGP neighbor, which was nearly simultaneously, just 0.002 seconds later.</p>
First message before reset received at Sep 12 16:42:16.573, sent at Sep 12 16:42:16.574	<p>Timestamp when the first message from a BGP neighbor was received by the local router before a reset occurred.</p> <p>In this specific output, the fields indicate the following:</p> <p><i>First message before reset received at Sep 12 16:42:16.573:</i> first message from the BGP neighbor was received at 16:42:16 on September 12th, before a reset occurred.</p> <p><i>sent at Sep 12 16:42:16.574:</i> Timestamp when the corresponding message was sent by the BGP neighbor, which was nearly simultaneous, just 0.001 seconds later.</p>
Max read throttled duration was 6769 msec starting at Sep 12 16:45:01.487, max InQ 1000 processed 930	<p>Maximum duration during which the read process was throttled, indicating a restriction or limitation on the rate of reading data.</p> <p>In this specific output, the fields indicate the following:</p> <p><i>Max read throttled duration was 6769 msec:</i> Maximum duration of throttling for reading data was 6769 milliseconds (approximately 6.769 seconds).</p> <p><i>starting at Sep 12 16:45:01.487:</i> Timestamp when this maximum throttling duration started, which was at 16:45:01 on September 12th.</p> <p><i>max InQ 1000 processed 930:</i> Maximum input queue (InQ) size was 1000, and during the throttled duration, 930 items were processed.</p>

Field	Description
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	<p>Maximum duration of a read throttle period on the network device.</p> <p>In this specific output, the fields indicate the following:</p> <p><i>Max read throttled duration before reset:</i> Maximum duration of the read throttle period, which was 5013 milliseconds or approximately 5.013 seconds.</p> <p><i>Starting at Sep 12 16:42:17.079:</i> Timestamp when the read throttle period started, which was at 16:42:17 on September 12th</p> <p><i>Max InQ 76 processed 0:</i> The segment <i>Max InQ 76</i> indicates that the maximum size of the input queue during the throttle period was 76. The segment <i>processed 0</i> indicates that no messages or data packets were processed during this throttle period.</p>
Max write throttled duration was 685 msec starting at Sep 12 16:45:08.486, max OutQ 1501 queued 57	<p>Maximum duration of the write throttle period, which was 685 milliseconds.</p> <p>In this specific output, the fields indicate the following:</p> <p><i>Max write throttled duration:</i> Maximum duration of the write throttle period, which was 685 milliseconds.</p> <p><i>Starting at Sep 12 16:45:08.486:</i> Timestamp when the write throttle period started, which was September 12th at 16:45:08.486.</p> <p><i>Max OutQ:</i> 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.</p> <p><i>Queued:</i> Number of items queued in the output queue during the throttle period. In this case, it was 57.</p>

Field	Description
Max write throttled duration before reset was 205 msec starting at Sep 12 16:42:21.849, max OutQ 1003 queued 1	<p>Maximum duration of a write throttle period on a network device before a reset occurred.</p> <p>In this specific output, the fields indicate the following:</p> <p><i>Max write throttled duration before reset:</i> Maximum duration of the write throttle period before a reset occurred, which was 205 milliseconds.</p> <p><i>Starting at Sep 12 16:42:21.849:</i> Timestamp when the write throttle period started, which was on September 12th at 16:42:21.849.</p> <p><i>Max OutQ:</i> 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.</p> <p><i>Queued:</i> Number of items queued in the output queue during the throttle period. In this case, it was 1.</p>
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

Table 61: Feature History Table

Feature Name	Release Name	Description
Enhanced Monitoring of BGP Memory Utilization	Release 24.4.1	<p>Introduced in this release on: Fixed Systems (8700) (select variants only*)</p> <p>*Enhanced Monitoring of BGP Memory Utilization is now supported on Cisco 8712-MOD-M routers.</p>

Feature Name	Release Name	Description
Enhanced Monitoring of BGP Memory Utilization	Release 24.2.11	<p>You can now enhance your network's reliability and efficiency with the Enhanced Monitoring of BGP Memory Utilization feature, allowing administrators to proactively oversee BGP's memory footprint—essential for routing within sophisticated networks. This feature acts as a vigilant alert system, offering regular assessments, documenting memory usage, and providing timely notifications as you approach critical memory thresholds, all to ensure optimal BGP memory management for uninterrupted network functionality.</p> <p>CLI:</p> <p>The feature modifies the output of the show command given below:</p> <ul style="list-style-type: none"> • show bgp memory history <p>YANG Data Model:</p> <ul style="list-style-type: none"> • Cisco-IOS-XR-ipv4-bgp-oper <p>(see GitHub, YANG Data Models Navigator)</p>

Overview

The Enhanced Monitoring of BGP Memory Utilization feature helps network administrators track memory usage by the BGP process within a network device. This feature is critical for maintaining the stability and performance of network operations, particularly in complex networks where BGP plays a central role in routing decisions.

Serving as an early warning system for potential memory-related issues within the BGP process, the Enhanced Monitoring of BGP Memory Utilization feature provides periodic checks, logs memory changes, and triggers notifications when critical thresholds are approached or exceeded. This proactive approach allows network administrators to manage BGP's memory usage effectively, thereby contributing to network reliability and stability.

Periodic Memory State Check

The feature incorporates an automated system that monitors the current memory state at regular intervals. The system performs checks every 60 seconds, a frequency designed to detect memory issues promptly while minimizing overhead.

BGP Trace and Circular Buffer Logging

When the memory state is checked, any changes are recorded in two locations. The BGP trace log maintains a detailed record of events and states within the BGP process, including memory usage. The circular buffer, a data structure with a fixed memory capacity, stores these logs; when it reaches full capacity, it overwrites the oldest entries with new ones. By logging memory changes in both places, the system ensures reliable information is available for troubleshooting and analysis.

Easily Understandable Show Outputs

The feature is designed to generate outputs (reports) that are clear and easy to understand. These reports offer a snapshot of current memory statistics, providing valuable insights for diagnosing issues.

Detection of Significant Memory Changes

The feature includes a built-in threshold to detect significant changes in memory usage. If the BGP process's memory utilization rises by more than 1% of the pre-configured resource limit compared to the last recorded report, this change is logged. This mechanism helps identify trends or sudden spikes in memory usage that could signal potential issues.

Resource Limit (rlimit)

The resource limit, or rlimit, is a predefined threshold that specifies the maximum amount of memory the BGP process can use. This limit is designed to prevent the BGP process from consuming excessive memory, which could negatively impact other processes and lead to system instability.

Syslog Notifications

As the memory usage by the BGP process reaches certain critical thresholds of the resource limit—specifically, 85%, 90%, and 95%—the system triggers notifications. These notifications are sent to the syslog, a service that records messages, which helps administrators take notice and action. These notifications are gradual, meaning they become more frequent as the memory usage gets closer to the resource limit, providing an escalating series of warnings that indicate the seriousness of the situation.

Verify Enhanced Monitoring of BGP Memory Utilization

Run the **show bgp memory history** command, to learn the history of memory utilization changes when memory utilization reaches a threshold of more than 1% of "rlimit". You can learn from the output details of memory usage, the history of memory utilization, percentage of resource limit utilized, memory variations between records, and calculations for networks, paths, path elements, and attributes, specifically for the default VRF.

In this example, the last line in the output given below represents the latest values. The value in the Memory(MB) column indicates the latest memory utilization (766 MB). The value in the Memory diff(MB) column (-136) is the difference between the memory utilization (901 MB) and the latest memory utilization (766 MB). The value in the Rlimit(%) column indicates the latest maximum memory utilization limit.

```
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 output given below displays the memory utilization history of the standby router.

```

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						

This table describes the significant fields shown in the display.

Table 62: Fields pertaining to Enhanced Monitoring of BGP Keepalive Messages in the output of show bgp neighbor 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.

Enhanced Next Hop Monitoring

Table 63: Feature History Table

Feature Name	Release Name	Description
Enhanced Next Hop Monitoring	Release 24.2.11	Introduced in this release on: Fixed Systems (8700) (select variants only*) *Enhanced Next Hop Monitoring is now supported on Cisco 8712-MOD-M routers.

Feature Name	Release Name	Description
Enhanced Next Hop Monitoring	Release 24.2.1	<p>You can now optimize routing decisions with the BGP Nexthop Event Tracking feature. This feature strengthens BGP's integration with the RIB by actively monitoring critical nexthop events. It enables you to swiftly pinpoint the root causes of network fluctuations, especially in large-scale deployments, by providing comprehensive event data.</p> <p>CLI:</p> <p>The feature modifies the output of the show command given below:</p> <ul style="list-style-type: none"> • show bgp nexthops <p>YANG Data Model:</p> <ul style="list-style-type: none"> • Cisco-IOS-XR-ipv4-bgp-oper <p>(see GitHub, YANG Data Models Navigator)</p>

Overview

Tracking Nexthop Changes BGP closely collaborates with the Routing Information Base (RIB) to monitor selected nexthops, receiving notifications for metric or reachability status changes. These triggers prompt BGP to recompute optimal routing paths for affected routes. During high routing activity or network instability, this process can strain CPU resources. The objective is to establish a robust monitoring system capturing events in detail, aiding administrators in pinpointing network variations' root causes, crucial in large-scale networks.

Detailed Reporting

To achieve comprehensive monitoring and analysis, the reporting system includes:

Event Counter Tracking:

Maintains counters for key event types per monitored nexthop, including unreachable, reachable, metric increase, and decrease.

Event History Logging:

Logs the last five events for each nexthop with timestamps for temporal context.

Recent Event Tracking:

Records details of nexthop transitions to reachable or unreachable states, along with precise timestamps.

Global Aggregation of Statistics:

Aggregate event counters globally by Address Family Identifier (AFI) for broader network health assessment.

Temporal Counter Analysis:

Continuously calculate and update event counters, reporting count variations within the last minute, three minutes, and five minutes, providing a dynamic network stability snapshot and routing decision efficacy evaluation.

View Enhanced Next Hop Monitoring

Run the **show bgp nexthops** command to view the details of nexthop reachability and metric change counters, offering insights into the network's routing dynamics and stability.

Router# **show bgp nexthops**

```
Fri Aug  4 18:28:02.428 IST
Total Nexthop Processing
  Time Spent: 0.009 secs
```

```
Maximum Nexthop Processing
  Received: 03:11:51
  Bestpaths Deleted: 0
  Bestpaths Changed: 110
  Time Spent: 0.001 secs
```

```
Last Notification Processing
  Received: 03:10:27
  Time Spent: 0.000 secs
```

```
Gateway Address Family: IPv4 Unicast
Table ID: 0xe0000000
Gateway Reference Count: 30
Gateway AF Bits : 0x991
Nexthop Count: 5
Critical Trigger Delay: 3000msec
Non-critical Trigger Delay: 10000msec
Nexthop Version: 3, RIB version: 3
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
0.0.0.0				25/3		
10.10.10.1	[R][C][NL]	0	e0000000	1/0	06:54:58 (Cri)	17/20
203.0.113.1	R][NC][NL]	2	e0000000	0/0	06:54:14 (Reg)	5/7
192.168.0.3	[R][NC][NL]	3	e0000000	0/3	04:10:07 (Non)	12/246
192.168.0.5	[R][NC][NL]	2	e0000000	1/1	06:54:21 (Non)	16/270

Counters

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

Run the **show bgp nexthops wide** to learn detailed information about BGP next-hop processing times, status codes, event counters, and metrics for each gateway address family. The event counter codes are used to provide additional status information about BGP next hops, followed by a table of specific next-hop entries with their respective statuses and metrics.

```
Router# show bgp nexthops wide
```

```
Fri Aug  4 19:26:16.845 IST
Total Nexthop Processing
  Time Spent: 0.009 secs
Maximum Nexthop Processing
  Received: 04:10:06
  Bestpaths Deleted: 0
  Bestpaths Changed: 110
  Time Spent: 0.001 secs

Last Notification Processing
  Received: 04:08:42
  Time Spent: 0.000 secs

Gateway Address Family: IPv4 Unicast
Table ID: 0xe0000000
Gateway Reference Count: 30
Gateway AF Bits : 0x991
Nexthop Count: 5
Critical Trigger Delay: 3000msec
Non-critical Trigger Delay: 10000msec
Nexthop Version: 3, RIB version: 3
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)

Event Counter Codes:

      R Reachable
      U Unreachable
      MI Metric Increased
      MD Metric Decreased
```

Next Hop	R	U	MI	MD	Status	Metric	Tbl-ID	Notf	LastRIBEvent	RefCount
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				

Run the **show bgp nexthops <nexthop>** to learn 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.

```

Router# show bgp nexthops 10.10.10.1
Wed Aug  2 16:20:02.058 IST
Nexthop: 10.10.10.1
  VRF: default
  Nexthop ID: 0x6000033, Version: 0
  Nexthop Flags: 0x00020002
  Nexthop Handle: 0x7fb2a07fafa4
  Tree Nexthop Handle: 0x7fb2a07fafa4
  Advertising neighbors:
    10.10.10.1
  RIB Related Information:
  Firsthop interface handle 0x01004020
    Gateway TBL Id: 0xe0000000    Gateway Flags: 0x00000080
    Gateway Handle: 0x286ae30
    Gateway: reachable, Connected route, prefix length 24
    Resolving Route: 10.10.10.0/24 (connected)
    Paths: 1
    RIB Nexthop ID: 0x6
    Nexthop sync slot: 23
    Status: [Reachable][Connected][Not Local] \
    Metric: 0
    ORR afi bits: 0x0
    Registration: Asynchronous, Completed: 00:10:32
    Events: Critical (1)/Non-critical (0)
    Last Received: 00:10:32 (Critical)
    Last gw update: (Crit-notif) 00:10:32(rib)
    Reference Count: 17
    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

  Prefix Related Information
    Active Tables: [IPv4 Unicast]
    Metrics: [0x0]
    Reference Counts: [17]
    Interface Handle: 0x0
    Attr ref-count: 20

```

This table describes the significant fields shown in the display.

Table 64: Fields pertaining to Enhanced Monitoring of BGP Keepalive Messages in the output of show bgp nexthops command

Field	Description
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.

Field	Description
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.

Enhanced Monitoring of Version-Rate Statistics

Table 65: Feature History Table

Feature Name	Release Name	Description
Enhanced Monitoring of Version-Rate Statistics	Release 24.2.11	<p>Introduced in this release on: Fixed Systems (8700) (select variants only*)</p> <p>*Enhanced Monitoring of Version-Rate Statistics is now supported on Cisco 8712-MOD-M routers.</p>
Enhanced Monitoring of Version-Rate Statistics	Release 24.2.1	<p>You can now effectively pinpoint the sources of BGP churn with the enhanced feature that monitors and sorts version bumps. This functionality empowers you to calculate version rates within defined intervals and organize them by their origin—whether reachable, unreachable, import, redistribution, or label-related—while maintaining a cumulative count of churn. The feature generates comprehensive reports on version bumps for both AFI and neighbor AFI levels, providing totals for each interval, bucketized categorization, and additional sources based on the main table's version number.</p> <p>CLI:</p> <p>The feature modifies the output of the show command given below:</p> <ul style="list-style-type: none">• <code>show bgp sessions version-rate</code> <p>YANG Data Model:</p> <ul style="list-style-type: none">• <code>Cisco-IOS-XR-ipv4-bgp-oper</code> <p>(see GitHub, YANG Data Models Navigator)</p>

Overview

The enhanced feature enables efficient identification of BGP churn sources by monitoring and sorting version bumps. It empowers users to calculate version rates within specified intervals and categorize them based on their origin, such as reachable, unreachable, import, redistribution, or label-related. Additionally, the feature maintains a cumulative churn count. Comprehensive reports on version bumps are generated for both AFI and neighbor AFI levels, including totals for each interval, bucketized categorization, and additional sources derived from the main table's version number.

Interval Mechanism Specification

The interval mechanism operates on fixed time frames rather than a sliding window approach. For instance, if an interval begins at 12:00 pm and ends at 12:30 pm, the subsequent data update occurs precisely at 12:30 pm, marking the start of the next interval from 12:30 pm to 01:00 pm.

NSR Synchronization Details

Both active and standby routers independently record their version bumps. Therefore, the standby router's version of the discussed commands may show unsynchronized values with the active router's statistics.

Reporting Data Parameters

The system now reports version bumps at both the Address-family identifier (AFI) level and the neighbor AFI level, providing totals and interval-specific rates. These are further categorized into reachable, unreachable, import, redistribute, and label buckets. Version bumps from other sources are also displayed, derived from the main table version number.

Verify Enhanced Monitoring of Version-Rate Statistics

Run the **show bgp sessions version-rate** command to get a detailed analysis of BGP version changes within specified intervals—Live, historical intervals 1, 2, and 3—highlighting their duration and timing. It categorizes these 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.

```
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
```

Run the **show bgp sessions version-rate live** command to learn about the real-time BGP session version-rate statistics, capturing changes within the most recent 5-minute interval across all BGP neighbors and address families, and includes metrics such as the total number of version changes along with reachability, speaker, AS number, queue sizes, and session states.

```
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      0
10.0.1.1       1      All          5      0      0      0      0
10.0.1.1       IPv4 Unicast    5      0      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 **show bgp sessions version-rate brief** command provides a concise overview of BGP version-rate statistics across multiple intervals, detailing the number of version changes for each BGP neighbor within the live interval and three historical periods, alongside session-specific information such as VRF, speaker ID, AS number, queue sizes, and state.

```

Router# show bgp sessions version-rate brief

```

```

Thu Nov  2 11:40:52.765 IST

```

```

Interval definition(s):

```

Interval	Duration (min)	Start time	End time
Live	5	Nov 2 11:37:22.029	Nov 2 11:40:53.072
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	Spk	AS	InQ	OutQ	NBRState	NSRState
10.10.10.1	default	0	200	0	0	Established	NSRReady
192.168.0.5	default	0	100	0	0	Established	NSRReady
10:10:10::1	default	0	200	0	0	Established	NSRReady
10.0.1.1	1	0	200	0	0	Established	NSRReady
10:0:1::1	1				0 200 0		

```

/* The output was too wide, so it was segmented; the below segment continues from above.
*/

```

Total	Live	Interval1	Interval2	Interval3
5	0	5	5	0
606	0	282	282	0
5	0	5	5	0
5	0	0	0	0

Run the **show bgp sessions version-rate brief live** command to get concise information only for the "Live" interval, which is typically the most recent 5-minute window. It does not show historical interval data but focuses exclusively on real-time or near real-time BGP version rate statistics. The "Live" interval data is the only interval column present next to the "Total" column, which shows the overall version bumps.

```
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	0	Established	NSRReady	5	0

This table describes the significant fields shown in the display.

Table 66: Fields pertaining to Enhanced Monitoring of Version-Rate Statistics in the output of show bgp sessions version-rate commands

Field	Description
Neighbor	The IP address or identifier of the BGP neighbor (another router or network device) being reported on.
VRF	Virtual Routing and Forwarding instance to which the BGP neighbor belongs.
Spk	Speaker ID, which is the identifier of the router from which the BGP session originates.
AS	Autonomous System number of the BGP neighbor.
InQ	The number of messages in the incoming queue waiting to be processed by the router for this neighbor.
OutQ	The number of messages in the outgoing queue waiting to be sent to the BGP neighbor.
NBRState	The current state of the BGP session with the neighbor (e.g., Established, Idle).
NSRState	Non-Stop Routing State, which indicates if the session is stable and capable of handling route refreshes without session reset.
Total	The total number of version changes across all intervals being reported.
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.
Live	The number of BGP version changes that occurred during the live (most recent) interval.
Interval1	The number of BGP version changes that occurred during the first historical interval.
Interval2	The number of BGP version changes that occurred during the second historical interval.
Interval3	The number of BGP version changes that occurred during the third historical interval.
Duration	Duration of the write throttle period in milliseconds.

Field	Description
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.

Flexible BGP Persistence

Table 67: Feature History Table

Feature Name	Release Name	Description
Flexible BGP Persistence	Release 24.3.1	

Feature Name	Release Name	Description
		<p>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])</p> <p>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.</p> <p>The feature introduces these changes:</p> <p>CLI:</p> <ul style="list-style-type: none"> • The default, any, and advertise-internal-only keywords are added to the long-lived-graceful-restart command. • The fields Long-lived Graceful Restart Stale Time Send Default, Default 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. <p>YANG Data Model:</p> <ul style="list-style-type: none"> • Cisco-IOS-XR-ipv4-bgp-cfg (see GitHub, YANG Data Models Navigator)

Flexible BGP Persistence and its Benefits

The Flexible BGP Persistence feature provides you the flexibility to enable LLGR without manual timeout configuration, advertise LLGR stale routes to non-LLGR eBGP neighbors seamlessly, and ensures controlled route distribution and policy enforcement by attaching NO_EXPORT community and setting local preference to 0.

- **Simplified configuration:** You can configure the router to enable and advertise the LLGR capability without explicitly setting a timeout value, and to accept the LLGR stale time provided by the peer without imposing its own threshold, by using the **long-lived-graceful-restart send default accept any** command.
- The **send default** keyword pair provides the flexibility from not having to specify a timeout value.

The **accept any** keyword pair allows the router to accept the LLGR stale time provided by the peer without imposing its own threshold for acceptance. This flexibility ensures compatibility with a wider range of peer configurations and simplifies the management of LLGR settings.

- **Enhanced network resilience:** This feature allows LLGR stale routes to be advertised to non-LLGR capable eBGP neighbors, enhancing network resilience. This capability ensures that even non-LLGR capable neighbors can benefit from the stability provided by LLGR, thereby contributing to overall network robustness.
- **Enhanced network stability:** This feature allows the attachment of the NO_EXPORT community and setting the local preference to 0 when advertising an LLGR route to an internal neighbor that has not negotiated the capability. The LLGR_STALE community is always sent along with NO_EXPORT community. The feature allows explicit configuration to enable this requirement through the **long-lived-graceful-restart capable advertise-internal-only** command.

This approach prevents the stale route from being propagated beyond the local Autonomous System (AS) and ensures that it is not preferred over other routes. This helps maintain network stability and prevents potential routing issues by ensuring that only capable and properly negotiated routes are used for forwarding traffic.

