



Cisco IOS XR Routing Configuration Guide for the Cisco CRS Router, Release 6.1.x

First Published: 2016-11-01

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

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: [www.cisco.com go trademarks](http://www.cisco.com/go/trademarks). Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2016 Cisco Systems, Inc. All rights reserved.



CONTENTS

PREFACE

Preface xxv

Changes to This Document xxv

Communications, Services, and Additional Information xxv

CHAPTER 1

New and Changed Routing Features 1

New and Changed Feature Information 1

CHAPTER 2

Implementing BGP 3

Prerequisites for Implementing BGP 6

Information About Implementing BGP 6

BGP Functional Overview 6

BGP Router Identifier 7

BGP Maximum Prefix - Discard Extra Paths 7

Restrictions 8

BGP Default Limits 8

BGP Next Hop Tracking 9

Next Hop as the IPv6 Address of Peering Interface 10

Scoped IPv4/VPNv4 Table Walk 11

Reordered Address Family Processing 11

New Thread for Next-Hop Processing 11

show, clear, and debug Commands 11

Autonomous System Number Formats in BGP 12

2-byte Autonomous System Number Format 12

4-byte Autonomous System Number Format 12

as-format Command 12

BGP Configuration 12

Configuration Modes	12
Neighbor Submode	18
Configuration Templates	18
Template Inheritance Rules	20
Viewing Inherited Configurations	24
No Default Address Family	29
Neighbor Address Family Combinations	29
Routing Policy Enforcement	29
Table Policy	31
Update Groups	32
BGP Update Generation and Update Groups	32
BGP Update Group	32
BGP Cost Community	32
How BGP Cost Community Influences the Best Path Selection Process	33
Cost Community Support for Aggregate Routes and Multipaths	34
Influencing Route Preference in a Multiexit IGP Network	35
BGP Cost Community Support for EIGRP MPLS VPN PE-CE with Back-door Links	36
Adding Routes to the Routing Information Base	37
BGP Best Path Algorithm	37
Comparing Pairs of Paths	37
Order of Comparisons	39
Best Path Change Suppression	40
Administrative Distance	41
Multiprotocol BGP	42
Route Dampening	44
Minimizing Flapping	44
BGP Routing Domain Confederation	45
BGP Route Reflectors	45
BGP Optimal Route Reflector	48
Use Case	49
RPL - if prefix is-best-path/is-best-multipath	52
Remotely Triggered Blackhole Filtering with RPL Next-hop Discard Configuration	53
Configuring Destination-based RTBH Filtering	53
Verification	55

Default Address Family for show Commands	55
TCP Maximum Segment Size	56
Per Neighbor TCP MSS	56
MPLS VPN Carrier Supporting Carrier	56
BGP Keychains	57
BGP Multicast VPN	57
Configuring an MDT Address Family Session in BGP	59
BGP Nonstop Routing	63
BGP Best-External Path	65
BGP Local Label Retention	66
BGP Over GRE Interfaces	66
Command Line Interface (CLI) Consistency for BGP Commands	66
BGP Additional Paths	66
iBGP Multipath Load Sharing	67
BGP Selective Multipath	68
Accumulated Interior Gateway Protocol Attribute	69
Per VRF and Per CE Label for IPv6 Provider Edge	70
Constrained Route Distribution for BGP/MPLS Internet Protocol VPNs	70
Constrained Route Distribution Benefits	70
BGP RT-constrain SAFI—rt-filter	70
Selective VRF Download	71
Line Card Roles and Filters in Selective VRF Download	72
BGP Accept Own	72
BGP DMZ Link Bandwidth for Unequal Cost Recursive Load Balancing	74
BFD Multihop Support for BGP	74
BGP Multi-Instance and Multi-AS	74
BGP Prefix Origin Validation Based on RPKI	75
Configuring RPKI Cache-server	75
Configuring RPKI Prefix Validation	78
Configure BGP Prefix Validation	79
Configuring RPKI Bestpath Computation	80
BGP 3107 PIC Updates for Global Prefixes	81
BGP Prefix Independent Convergence for RIB and FIB	82
BGP Update Message Error Handling	83

BGP Attribute Filtering	83
BGP Attribute Filter Actions	83
BGP Error Handling and Attribute Filtering Syslog Messages	84
BGP Link-State	84
BGP Permanent Network	84
BGP VRF Dynamic Route Leaking	85
Resilient Per-CE Label Mode	86
BGP Multipath Enhancements	86
MVPN with BGP SAFI-2 and SAFI-129	87
How to Implement BGP	88
Enabling BGP Routing	88
Configuring Multiple BGP Instances for a Specific Autonomous System	90
Configuring a Routing Domain Confederation for BGP	91
Resetting an eBGP Session Immediately Upon Link Failure	92
Logging Neighbor Changes	92
Adjusting BGP Timers	92
Changing the BGP Default Local Preference Value	93
Configuring the MED Metric for BGP	94
Configuring BGP Weights	95
Tuning the BGP Best-Path Calculation	96
Indicating BGP Back-door Routes	97
Configuring Aggregate Addresses	98
Redistributing iBGP Routes into IGP	99
Configuring Discard Extra Paths	100
Configuring Per Neighbor TCP MSS	101
Disabling Per Neighbor TCP MSS	103
Redistributing Prefixes into Multiprotocol BGP	105
Configuring BGP Route Dampening	106
Applying Policy When Updating the Routing Table	111
Setting BGP Administrative Distance	112
Configuring a BGP Neighbor Group and Neighbors	113
Configuring a Route Reflector for BGP	115
Configuring BGP Route Filtering by Route Policy	117
Configuring BGP Attribute Filtering	118

Configuring BGP Next-Hop Trigger Delay	119
Disabling Next-Hop Processing on BGP Updates	120
Configuring BGP Community and Extended-Community Advertisements	121
Configuring the BGP Cost Community	123
Configuring Software to Store Updates from a Neighbor	126
BGP Persistence	127
BGP Persistence Configuration: Example	128
BGP Graceful Maintenance	128
Restrictions for BGP Graceful Maintenance	129
Graceful Maintenance Operation	129
Inter Autonomous System	130
No Automatic Shutdown	130
When to Shut Down After Graceful Maintenance	130
Activate Graceful Maintenance under BGP Router (All Neighbors)	130
Direct Router to Reduce Route Preference	134
Bring Router or Link Back into Service	135
Show Command Outputs to Verify BGP Graceful Maintenance	135
L3VPN iBGP PE-CE	136
L3VPN iBGP PE-CE Overview	137
Restrictions for L3VPN iBGP PE-CE	137
Configuring L3VPN iBGP PE-CE	138
Flow-tag propagation	140
Restrictions for flow-tag propagation	140
Source and destination-based flow tag	141
Configure Source and Destination-based Flow Tag	141
Configuring a VPN Routing and Forwarding Instance in BGP	142
Defining Virtual Routing and Forwarding Tables in Provider Edge Routers	142
Configuring the Route Distinguisher	144
Configuring BGP to Advertise VRF Routes for Multicast VPN from PE to PE	146
Advertising VRF Routes for MVPNv4 from PE to PE	146
Advertising VRF Routes for MVPNv6 from PE to PE	150
Configuring PE-PE or PE-RR Interior BGP Sessions	154
Configuring Route Reflector to Hold Routes That Have a Defined Set of RT Communities	156
Configuring BGP as a PE-CE Protocol	157

Redistribution of IGPs to BGP	161
Configuring Keychains for BGP	163
Configuring an MDT Address Family Session in BGP	164
Disabling a BGP Neighbor	168
Resetting Neighbors Using BGP Inbound Soft Reset	169
Resetting Neighbors Using BGP Outbound Soft Reset	170
Resetting Neighbors Using BGP Hard Reset	170
Clearing Caches, Tables, and Databases	171
Displaying System and Network Statistics	172
Displaying BGP Process Information	174
Monitoring BGP Update Groups	175
Configuring BGP Nonstop Routing	176
Configuring Best-External Path Advertisement	177
Installing Primary Backup Path for Prefix Independent Convergence (PIC)	177
Retaining Allocated Local Label for Primary Path	179
Configuring BGP Additional Paths	179
Configuring iBGP Multipath Load Sharing	181
Originating Prefixes with AiGP	182
Configuring BGP Accept Own	183
Configuring BGP Link-State	184
Configuring BGP Link-state	184
Configuring Domain Distinguisher	185
Configuring BGP Permanent Network	186
Configuring BGP Permanent Network	186
How to Advertise Permanent Network	188
Enabling BGP Unequal Cost Recursive Load Balancing	189
Configuring VRF Dynamic Route Leaking	190
Configuring Resilient Per-CE Label Mode	192
Configuring Resilient Per-CE Label Mode Under VRF Address Family	192
Configuring Resilient Per-CE Label Mode Using a Route-Policy	194
Configuration Examples for Implementing BGP	195
Enabling BGP: Example	195
Displaying BGP Update Groups: Example	197
BGP Neighbor Configuration: Example	197

BGP Confederation: Example	198
BGP Route Reflector: Example	199
BGP MDT Address Family Configuration: Example	200
BGP Nonstop Routing Configuration: Example	200
Best-External Path Advertisement Configuration: Example	200
Primary Backup Path Installation: Example	201
Allocated Local Label Retention: Example	201
iBGP Multipath Loadsharing Configuration: Example	201
Discard Extra Paths Configuration: Example	201
Displaying Discard Extra Paths Information: Example	202
Configure Per Neighbor TCP MSS: Examples	202
Verify Per Neighbor TCP MSS: Examples	204
Originating Prefixes With AiGP: Example	206
BGP Accept Own Configuration: Example	207
BGP Unequal Cost Recursive Load Balancing: Example	208
VRF Dynamic Route Leaking Configuration: Example	210
Flow-tag propagation	210
Restrictions for Flow-Tag Propagation	211
Where to Go Next	211
Additional References	211
<hr/>	
CHAPTER 3	Implementing BGP Flowspec 215
	BGP Flow Specification 215
	Limitations 216
	BGP Flowspec Conceptual Architecture 216
	Information About Implementing BGP Flowspec 217
	Flow Specifications 217
	Supported Matching Criteria and Actions 218
	Traffic Filtering Actions 221
	BGP Flowspec Client-Server (Controller) Model and Configuration with ePBR 222
	Configuring BGP Flowspec with ePBR 224
	Enable BGP Flowspec 224
	Configure a Class Map 225
	Configure a Policy Map 227

Link BGP Flowspec to ePBR Policies	229
Verify BGP Flowspec	232
Preserving Redirect Nexthop	234
Validate BGP Flowspec	235
Disabling BGP Flowspec	236
Disable Flowspec Redirect and Validation	237
Configuration Examples for Implementing BGP Flowspec	238
Flowspec Rule Configuration	238
Drop Packet Length	239
Redirect traffic and rate-limit: Example	240
Redirect Traffic from Global to VRF (vrf1)	240
Remark DSCP	240
Additional References for BGP Flowspec	240

CHAPTER 4**Implementing BFD 243**

Prerequisites for Implementing BFD	247
Restrictions for Implementing BFD	248
Information About BFD	249
Differences in BFD in Cisco IOS XR Software and Cisco IOS Software	249
BFD Modes of Operation	249
BFD Packet Information	250
BFD Source and Destination Ports	250
BFD Packet Intervals and Failure Detection	250
Priority Settings for BFD Packets	254
BFD for IPv4	255
BFD for IPv6	256
BFD on Bundled VLANs	256
BFD Over Member Links on Link Bundles	257
Overview of BFD State Change Behavior on Member Links and Bundle Status	258
BFD Multipath Sessions	260
BFD for MultiHop Paths	260
Setting up BFD Multihop	261
Limitations of BFD	261
BFD over MPLS Traffic Engineering LSPs	261

Echo Timer configuration for BFD on Bundle Interfaces	262
Bidirectional Forwarding Detection over Logical Bundle	263
BFD IPv6 Multihop	263
BFD over Pseudowire Headend	264
BFD over Satellite Interfaces	264
Limitations	264
BFD over Bundles CISCO/IETF Mode Support on a Per Bundle Basis	265
BFD Object Tracking	265
How to Configure BFD	266
BFD Configuration Guidelines	266
Configuring BFD Under a Dynamic Routing Protocol or Using a Static Route	266
Enabling BFD on a BGP Neighbor	266
Enabling BFD for OSPF on an Interface	268
Enabling BFD for OSPFv3 on an Interface	269
Enabling BFD on a Static Route	271
Configuring BFD on Bundle Member Links	272
Prerequisites for Configuring BFD on Bundle Member Links	272
Specifying the BFD Destination Address on a Bundle	272
Enabling BFD Sessions on Bundle Members	273
Configuring the Minimum Thresholds for Maintaining an Active Bundle	274
Configuring BFD Packet Transmission Intervals and Failure Detection Times on a Bundle	275
Configuring Allowable Delays for BFD State Change Notifications Using Timers on a Bundle	276
Configure BFD over Bundles CISCO/IETF Mode Support on a Per Bundle Basis	277
Enabling Echo Mode to Test the Forwarding Path to a BFD Peer	279
Overriding the Default Echo Packet Source Address	279
Specifying the Echo Packet Source Address Globally for BFD	280
Specifying the Echo Packet Source Address on an Individual Interface or Bundle	280
Configuring BFD Session Teardown Based on Echo Latency Detection	281
Delaying BFD Session Startup Until Verification of Echo Path and Latency	282
Disabling Echo Mode	283
Disabling Echo Mode on a Router	283
Disabling Echo Mode on an Individual Interface or Bundle	284
Minimizing BFD Session Flapping Using BFD Dampening	285

Enabling and Disabling IPv6 Checksum Support	285
Enabling and Disabling IPv6 Checksum Calculations for BFD on a Router	286
Enabling and Disabling IPv6 Checksum Calculations for BFD on an Individual Interface or Bundle	286
Clearing and Displaying BFD Counters	287
Configuring Coexistence Between BFD over Bundle (BoB) and BFD over Logical Bundle (BLB)	288
Configuring BFD over MPLS Traffic Engineering LSPs	289
Enabling BFD Parameters for BFD over TE Tunnels	289
Configuring BFD Bring up Timeout	290
Configuring BFD Dampening for TE Tunnels	291
Configuring Periodic LSP Ping Requests	292
Configuring BFD at the Tail End	293
Configuring BFD over LSP Sessions on Line Cards	294
Configuring BFD Object Tracking:	295
Configuration Examples for Configuring BFD	296
BFD Over BGP: Example	296
BFD Over OSPF: Examples	296
BFD Over Static Routes: Examples	297
BFD on Bundled VLANs: Example	297
BFD Over Bridge Group Virtual Interface: Example	298
BFD on Bundle Member Links: Examples	300
Echo Packet Source Address: Examples	302
Echo Latency Detection: Examples	302
Echo Startup Validation: Examples	303
BFD Echo Mode Disable: Examples	303
BFD Dampening: Examples	304
BFD IPv6 Checksum: Examples	304
BFD Peers on Routers Running Cisco IOS and Cisco IOS XR Software: Example	305
BFD over MPLS TE LSPs: Examples	305
BFD over MPLS TE Tunnel Head-end Configuration: Example	305
BFD over MPLS TE Tunnel Tail-end Configuration: Example	306
Where to Go Next	306
Additional References	306

Related Documents	306
Standards	306
RFCs	307
MIBs	307
Technical Assistance	307

CHAPTER 5
Implementing EIGRP 309

Prerequisites for Implementing EIGRP	310
Restrictions for Implementing EIGRP	310
Information About Implementing EIGRP	310
EIGRP Functional Overview	310
EIGRP Features	311
EIGRP Components	311
EIGRP Configuration Grouping	312
EIGRP Configuration Modes	312
EIGRP Interfaces	314
Redistribution for an EIGRP Process	314
Metric Weights for EIGRP Routing	314
Mismatched K Values	314
Goodbye Message	315
Percentage of Link Bandwidth Used for EIGRP Packets	316
Floating Summary Routes for an EIGRP Process	316
Split Horizon for an EIGRP Process	318
Adjustment of Hello Interval and Hold Time for an EIGRP Process	318
Stub Routing for an EIGRP Process	318
Route Policy Options for an EIGRP Process	319
EIGRP Layer 3 VPN PE-CE Site-of-Origin	320
Router Interoperation with the Site-of-Origin Extended Community	320
Route Manipulation using SoO match condition	321
IPv6 and IPv6 VPN Provider Edge Support over MPLS and IP	322
EIGRP v4/v6 Authentication Using Keychain	323
EIGRP Wide Metric Computation	323
EIGRP Support for BFD	323
How to Implement EIGRP	323

Enabling EIGRP Routing	324
Configuring Route Summarization for an EIGRP Process	325
Redistributing Routes for EIGRP	327
Creating a Route Policy and Attaching It to an EIGRP Process	328
Configuring Stub Routing for an EIGRP Process	330
Configuring EIGRP as a PE-CE Protocol	331
Redistributing BGP Routes into EIGRP	332
Monitoring EIGRP Routing	334
Configuring an EIGRP Authentication Keychain	336
Configuring an Authentication Keychain for an IPv4/IPv6 Interface on a Default VRF	336
Configuring an Authentication Keychain for an IPv4/IPv6 Interface on a Nondefault VRF	337
Configuring unicast neighbors	338
Remote Neighbor Session Policy	339
Understanding Neighbor Terms	340
Remote Unicast-Listen (Point-to-Point) Neighbors	340
Restrictions for remote neighbors	341
Inheritance and precedence of the remote neighbor configurations	341
How to configure remote unicast neighbors	341
Configuration Examples for Implementing EIGRP	343
Configuring a Basic EIGRP Configuration: Example	343
Configuring an EIGRP Stub Operation: Example	343
Configuring an EIGRP PE-CE Configuration with Prefix-Limits: Example	344
Configuring an EIGRP Authentication Keychain: Example	344
Additional References	345

CHAPTER 6**Implementing IS-IS 347**

Prerequisites for Implementing IS-IS	347
Implementing IS-IS	347
Configuration Examples for Implementing IS-IS	348
Configuring Single-Topology IS-IS for IPv6: Example	348
Configuring Multitopology IS-IS for IPv6: Example	348
Redistributing IS-IS Routes Between Multiple Instances: Example	348
Tagging Routes: Example	349
Configuring IS-IS Overload Bit Avoidance: Example	350

Where to Go Next	350
Additional References	350

CHAPTER 7**Implementing OSPF 353**

Prerequisites for Implementing OSPF	355
Information About Implementing OSPF	356
OSPF Functional Overview	356
Key Features Supported in the Cisco IOS XR Software OSPF Implementation	357
Comparison of Cisco IOS XR Software OSPFv3 and OSPFv2	358
OSPF Hierarchical CLI and CLI Inheritance	358
OSPF Routing Components	359
Autonomous Systems	359
Areas	359
Routers	360
OSPF Process and Router ID	361
Supported OSPF Network Types	362
Route Authentication Methods for OSPF	362
Plain Text Authentication	362
MD5 Authentication	362
Authentication Strategies	362
Key Rollover	363
Neighbors and Adjacency for OSPF	363
OSPF strict-mode Support for BFD Dampening	363
Enabling strict-mode	363
BFD strict-mode: Example	364
OSPF FIB Download Notification	365
Designated Router (DR) for OSPF	366
Default Route for OSPF	366
Link-State Advertisement Types for OSPF Version 2	366
Link-State Advertisement Types for OSPFv3	367
Virtual Link and Transit Area for OSPF	368
Passive Interface	369
OSPFv2 Sham Link Support for MPLS VPN	369
OSPF SPF Prefix Prioritization	371

Route Redistribution for OSPF	373
OSPF Shortest Path First Throttling	373
Nonstop Forwarding for OSPF Version 2	374
Graceful Shutdown for OSPFv3	374
Modes of Graceful Restart Operation	375
Graceful Restart Requirements and Restrictions	376
Warm Standby and Nonstop Routing for OSPF Version 2	377
Warm Standby for OSPF Version 3	377
Multicast-Intact Support for OSPF	377
Load Balancing in OSPF Version 2 and OSPFv3	378
Multi-Area Adjacency for OSPF Version 2	378
Label Distribution Protocol IGP Auto-configuration for OSPF	379
OSPF Authentication Message Digest Management	379
GTSM TTL Security Mechanism for OSPF	380
Path Computation Element for OSPFv2	380
OSPF Queue Tuning Parameters	380
OSPF IP Fast Reroute Loop Free Alternate	381
OSPF Over GRE Interfaces	381
Management Information Base (MIB) for OSPFv3	381
VRF-lite Support for OSPFv2	381
OSPFv3 Timers Link-state Advertisements and Shortest Path First Throttle Default Values Update	382
Unequal Cost Multipath Load-balancing for OSPF	382
How to Implement OSPF	383
Enabling OSPF	383
Configuring Stub and Not-So-Stubby Area Types	384
Configuring Neighbors for Nonbroadcast Networks	386
Configuring Authentication at Different Hierarchical Levels for OSPF Version 2	389
Controlling the Frequency That the Same LSA Is Originated or Accepted for OSPF	392
Creating a Virtual Link with MD5 Authentication to Area 0 for OSPF	393
Examples	396
Summarizing Subnetwork LSAs on an OSPF ABR	397
Redistribute Routes into OSPF	398
Configuring OSPF Shortest Path First Throttling	401

Examples	402
Configuring Nonstop Forwarding Specific to Cisco for OSPF Version 2	403
Configuring OSPF Version 2 for MPLS Traffic Engineering	405
Examples	406
Configuring OSPFv3 Graceful Restart	408
Displaying Information About Graceful Restart	409
Configuring an OSPFv2 Sham Link	410
Enabling Nonstop Routing for OSPFv2	413
Enabling Nonstop Routing for OSPFv3	414
Configuring OSPF SPF Prefix Prioritization	414
Enabling Multicast-intact for OSPFv2	416
Associating Interfaces to a VRF	416
Configuring OSPF as a Provider Edge to Customer Edge (PE-CE) Protocol	417
Creating Multiple OSPF Instances (OSPF Process and a VRF)	419
Configuring Multi-area Adjacency	421
Configuring Label Distribution Protocol IGP Auto-configuration for OSPF	422
Configuring LDP IGP Synchronization: OSPF	423
Configuring Authentication Message Digest Management for OSPF	424
Examples	425
Configuring Generalized TTL Security Mechanism (GTSM) for OSPF	426
Examples	428
Verifying OSPF Configuration and Operation	428
Configuring OSPF Queue Tuning Parameters	430
Configuring IP Fast Reroute Loop-free Alternate	431
Enabling IPFRR LFA	431
Excluding an Interface From IP Fast Reroute Per-link Computation	432
Enabling OSPF Interaction with SRMS Server	433
Configuration Examples for Implementing OSPF	435
Cisco IOS XR Software for OSPF Version 2 Configuration: Example	435
CLI Inheritance and Precedence for OSPF Version 2: Example	436
MPLS TE for OSPF Version 2: Example	437
ABR with Summarization for OSPFv3: Example	437
ABR Stub Area for OSPFv3: Example	438
ABR Totally Stub Area for OSPFv3: Example	438

Configuring OSPF SPF Prefix Prioritization: Example	438
Route Redistribution for OSPFv3: Example	439
Virtual Link Configured Through Area 1 for OSPFv3: Example	440
Virtual Link Configured with MD5 Authentication for OSPF Version 2: Example	440
VPN Backbone and Sham Link Configured for OSPF Version 2: Example	441
OSPF Queue Tuning Parameters Configuration: Example	443
Where to Go Next	444
Additional References	444

CHAPTER 8

Implementing IP Fast Reroute Loop-Free Alternate	447
Prerequisites for IPv4/IPv6 Loop-Free Alternate Fast Reroute	447
Restrictions for Loop-Free Alternate Fast Reroute	447
IS-IS and IP FRR	448
Repair Paths	448
LFA Overview	449
LFA Calculation	449
Interaction Between RIB and Routing Protocols	449
Configuring Fast Reroute Support	450
Configuring IPv4 Loop-Free Alternate Fast Reroute Support: Example	452
Additional References	452

CHAPTER 9

Implementing and Monitoring RIB	455
Prerequisites for Implementing RIB	456
Information About RIB Configuration	456
Overview of RIB	456
RIB Data Structures in BGP and Other Protocols	456
RIB Administrative Distance	457
RIB Support for IPv4 and IPv6	457
RIB Statistics	458
IPv6 Provider Edge IPv6 and IPv6 VPN Provider Edge Transport over MPLS	458
IP Fast Reroute	458
RIB Quarantining	459
Route and Label Consistency Checker	459
Flex-LSR Label Switch Processor 140	460

How to Deploy and Monitor RIB	460
Verifying RIB Configuration Using the Routing Table	460
Verifying Networking and Routing Problems	461
Disabling RIB Next-hop Dampening	462
Configuring RCC and LCC	463
Enabling RCC and LCC On-demand Scan	463
Enabling RCC and LCC Background Scan	464
Configuration Examples for RIB Monitoring	465
Output of show route Command: Example	466
Output of show route backup Command: Example	466
Output of show route best-local Command: Example	466
Output of show route connected Command: Example	467
Output of show route local Command: Example	467
Output of show route longer-prefixes Command: Example	467
Output of show route next-hop Command: Example	467
Enabling RCC and LCC: Example	468
Where to Go Next	468
Additional References	469

CHAPTER 10
Implementing RIP 471

Prerequisites for Implementing RIP	472
Information About Implementing RIP	472
RIP Functional Overview	472
Split Horizon for RIP	473
Route Timers for RIP	473
Route Redistribution for RIP	473
Default Administrative Distances for RIP	474
Routing Policy Options for RIP	475
Authentication Using Keychain in RIP	475
In-bound RIP Traffic on an Interface	476
Out-bound RIP Traffic on an Interface	477
How to Implement RIP	477
Enabling RIP	477
Customizing RIP	479

Control Routing Information	480
Creating a Route Policy for RIP	482
Configuring RIP Authentication Keychain	483
Configuring RIP Authentication Keychain for IPv4 Interface on a Non-default VRF	483
Configuring RIP Authentication Keychain for IPv4 Interface on Default VRF	485
Configuration Examples for Implementing RIP	486
Configuring a Basic RIP Configuration: Example	486
Configuring RIP on the Provider Edge: Example	486
Adjusting RIP Timers for each VRF Instance: Example	486
Configuring Redistribution for RIP: Example	487
Configuring Route Policies for RIP: Example	488
Configuring Passive Interfaces and Explicit Neighbors for RIP: Example	488
Controlling RIP Routes: Example	489
Configuring RIP Authentication Keychain: Example	489
Additional References	489

CHAPTER 11
Implementing Routing Policy 491

Prerequisites for Implementing Routing Policy	493
Restrictions for Implementing Routing Policy	493
Information About Implementing Routing Policy	493
Routing Policy Language	493
Routing Policy Language Overview	494
Routing Policy Language Structure	494
Routing Policy Language Components	503
Routing Policy Language Usage	503
Routing Policy Configuration Basics	505
Policy Definitions	505
Parameterization	506
Parameterization at Attach Points	507
Global Parameterization	508
Semantics of Policy Application	508
Boolean Operator Precedence	508
Multiple Modifications of the Same Attribute	509
When Attributes Are Modified	510

Default Drop Disposition	510
Control Flow	510
Policy Verification	511
Policy Statements	513
Remark	513
Disposition	513
Action	515
If	515
Boolean Conditions	516
apply	518
Attach Points	518
BGP Policy Attach Points	519
OSPF Policy Attach Points	542
OSPFv3 Policy Attach Points	546
IS-IS Policy Attach Points	548
EIGRP Policy Attach Points	550
RIP Policy Attach Points	554
PIM Policy Attach Points	557
Nondestructive Editing of Routing Policy	558
Attached Policy Modification	558
Nonattached Policy Modification	559
Editing Routing Policy Configuration Elements	559
Hierarchical Policy Conditions	561
Apply Condition Policies	561
Nested Wildcard Apply Policy	564
Wildcards for Route Policy Sets	565
Use Wildcards For Routing Policy Sets	565
VRF Import Policy Enhancement	569
Flexible L3VPN Label Allocation Mode	569
Match Aggregated Route	570
Remove Private AS in Inbound Policy	570
Set Administrative Distance	570
How to Implement Routing Policy	570
Defining a Route Policy	570

Attaching a Routing Policy to a BGP Neighbor	571
Modifying a Routing Policy Using a Text Editor	572
Configuration Examples for Implementing Routing Policy	573
Routing Policy Definition: Example	573
Simple Inbound Policy: Example	574
Modular Inbound Policy: Example	575
Use Wildcards For Routing Policy Sets	576
Translating Cisco IOS Route Maps to Cisco IOS XR Routing Policy Language: Example	580
VRF Import Policy Configuration: Example	580
Additional References	580

CHAPTER 12**Implementing Static Routes 583**

Prerequisites for Implementing Static Routes	584
Restrictions for Implementing Static Routes	584
Information About Implementing Static Routes	584
Static Route Functional Overview	584
Default Administrative Distance	585
Directly Connected Routes	585
Recursive Static Routes	585
Fully Specified Static Routes	586
Floating Static Routes	586
Default VRF	586
IPv4 and IPv6 Static VRF Routes	587
How to Implement Static Routes	587
Configure Static Route	587
Configuring a Static Route Under Multicast SAFI	588
Configure Floating Static Route	590
Configure Static Routes Between PE-CE Routers	591
Change Maximum Number of Allowable Static Routes	592
Associate VRF with a Static Route	593
Configuration Examples	594
Configuring Traffic Discard: Example	595
Configuring a Fixed Default Route: Example	595
Configuring a Floating Static Route: Example	595

Configure Native UCMP for Static Routing	595
Configuring a Static Route Between PE-CE Routers: Example	597
Where to Go Next	597
Additional References	597

CHAPTER 13**Implementing RCMD 599**

Route Convergence Monitoring and Diagnostics	599
Route Convergence Monitoring and Diagnostics Database Monitoring	600
Configuring Route Convergence Monitoring and Diagnostics	600
Route Convergence Monitoring and Diagnostics Prefix Monitoring	603
Route Convergence Monitoring and Diagnostics OSPF Type 3/5/7 Link-state Advertisements Monitoring	603
Enabling RCMD Monitoring for IS-IS Prefixes	603
Enable RCMD Monitoring for OSPF Prefixes	604
Enabling RCMD Monitoring for Type 3/5/7 OSPF LSAs	605
Enabling RCMD Monitoring for IS-IS Prefixes: Example	606
Enabling RCMD Monitoring for OSPF Prefixes: Example	606
Enabling RCMD Monitoring for Type 3/5/7 OSPF LSAs: Example	607

CHAPTER 14**Implementing Data Plane Security 609**

Information about Data Plane Security	609
Source RLOC Decapsulation Filtering	609
EID Instance Membership Distribution	610
Map-Server Membership Gleaning and Distribution	611
Decapsulation Filtering on (P)xTRs	613
TCP-based Reliable Transport Sessions	614
How to Implement Data Plane Security	614
Enable Source RLOC-based Decapsulation Filtering	614
Create, Maintain and Distribute Decapsulation Filter Lists	618
Add or Override Decapsulation Filter List	619
Reset LISP TCP Reliable Transport Session	620
Verify Data Plane Security Configurations	620
Additional References	624



Preface

The *Routing Configuration Guide for Cisco CRS Routers* preface contains these sections:

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

Changes to This Document

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

Table 1: Changes to This Document

Date	Summary
November 2016	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 impact 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 Marketplace](#).
- 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 Routing Features

This table summarizes the new and changed feature information for the *Routing Configuration Guide for Cisco CRS Routers*, and tells you where they are documented.

- [New and Changed Feature Information, on page 1](#)

New and Changed Feature Information

Table 2: New and Changed Features

Feature	Description	Changed in Release	Where Documented
BGP Optimal Route Reflector	This feature was introduced.	Release 6.1.2	BGP Optimal Route Reflector



CHAPTER 2

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 the conceptual and configuration information for BGP on Cisco IOS XR software.



Note For more information about BGP on the Cisco IOS XR software and complete descriptions of the BGP commands listed in this module, see [Related Documents, on page 211](#) section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing BGP

Release	Modification
Release 2.0	This feature was introduced.
Release 3.0	No modification.
Release 3.2	No modification.
Release 3.3.0	VPN routing and forwarding (VRF) support was added, including information on VRF command modes and command syntax. BGP cost community information was added.
Release 3.4.0	The following features were supported: <ul style="list-style-type: none">• Four-byte autonomous system (AS) number• Carrier supporting carrier (CSC) for BGP was added. See <i>Cisco IOS XR Multiprotocol Label Switching Protocol Configuration Guide</i> for information• Key chains

Release	Modification
Release 3.5.0	<p>The following features were supported:</p> <ul style="list-style-type: none"> • IPv6 Provider Edge and IPv6 VPN Provider Edge over Multiprotocol Label Switching • Neighbor-specific VRF IPv6 address family configurations • Address family group-specific VPNv6 configurations • VPN4/VPNv6 over IP core using L2TPv3 tunnels • Multicast Distribution Tree (MDT) Subaddress Family Identifier Information (SAFI) support for multicast VPN (MVPN)
Release 3.6.0	No modification.
Release 3.7.0	<p>The following features were supported:</p> <ul style="list-style-type: none"> • Advertisement of VRF routes for multicast VPNs (MVPN) for both IPv4 and IPv6 address families from PE to PE • Edits were made to existing MVPN procedures based on new support for IPv6 multicast VPNs • Procedure Configuring an MDT Address Family Session in BGP, on page 59 was updated to reflect MVPN configuration of MDT SAFI from PE to PE
Release 3.8.0	<p>The following features were supported:</p> <ul style="list-style-type: none"> • Border Gateway Protocol (BGP) nonstop routing (NSR) with stateful switchover (SSO) • Next hop as the IPv6 address of peering interface • Reset weight on import of VPN routes • New commands enforce-first-as and enforce-first-as-disable were introduced to provide enable and disable configuration options for enforce-first-as feature in Neighbor, Neighbor group, and Session group configuration modes.
Release 3.9.0	<p>The following features were supported:</p> <ul style="list-style-type: none"> • BGP Best-External Path • BGP Prefix Independent Convergence Unipath Primary Backup • BGP Local Label Retention • BGP Over GRE Interfaces • Asplain notation for 4-byte Autonomous System Number • Command Line Interface (CLI) consistency for BGP commands • L2VPN Address Family Configuration Mode

Release	Modification
Release 4.0.0	The following features were supported: <ul style="list-style-type: none"> • Accumulated iGP (AiGP) • BGP Add Path Advertisement • iBGP Multipath Load Sharing • Next Hop Self on Route Reflector for iBGP+Label
Release 4.1.0	The following features were supported: <ul style="list-style-type: none"> • BGP RT Constrained Route Distribution
Release 4.1.1	The BGP Accept Own feature was added.
Release 4.2.0	The following features were supported: <ul style="list-style-type: none"> • BGP Multi-Instance/Multi-AS Support • BFD Multihop Support for BGP • BGP Error Handling Support for Distributed BGP (bgp distributed speaker) configuration was removed.
Release 4.2.1	The following features were supported: <ul style="list-style-type: none"> • BGP 3107 PIC Updates for Global Prefixes • BGP Prefix Independent Convergence for RIB and FIB • BGP Prefix Origin Validation Based on RPKI
Release 4.2.3	The BGP Attribute Filtering feature was added.
Release 4.3.0	The BGP DMZ Link Bandwidth for Unequal Cost Recursive Load Balancing feature was added.
Release 4.3.1	The following features were supported <ul style="list-style-type: none"> • BGP VRF Dynamic Route Leaking The label-allocation-mode command is renamed the label mode command.
Release 5.3.1	The following features were supported: <ul style="list-style-type: none"> • L3VPN iBGP-PE-CE configuration • Source-based flow tag • Discard extra paths

Release	Modification
Release 5.3.2	<p>The following features were supported:</p> <ul style="list-style-type: none"> • Graceful Maintenance • Per Neighbor TCP MSS

- [Prerequisites for Implementing BGP, on page 6](#)
- [Information About Implementing BGP, on page 6](#)
- [How to Implement BGP, on page 88](#)
- [Configuration Examples for Implementing BGP, on page 195](#)
- [Flow-tag propagation, on page 210](#)
- [Where to Go Next, on page 211](#)
- [Additional References, on page 211](#)

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.

Information About Implementing BGP

To implement BGP, you need to understand the following concepts:

BGP Functional Overview

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 Other than enabling RTC (route target constraint) with `address-family ipv4 rtfiler` command, there is no separate configuration needed to enable RTC for BGP EVPN.



Note For information on configuring BGP to distribute Multiprotocol Label Switching (MPLS) Layer 3 virtual private network (VPN) information, see the Cisco IOS XR Multiprotocol Label Switching Configuration Guide for the Cisco CRS-1 Router.

For information on BGP support for Bidirectional Forwarding Detection (BFD), see the *Cisco IOS XR Interface and Hardware Configuration Guide for the Cisco CRS-1 Router* and the *Cisco IOS XR Interface and Hardware Command Reference for the Cisco CRS-1 Router*.

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 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. Default limits are used if the user does not configure the maximum number of prefixes for the address family. For default limits, refer to [BGP Default Limits, on page 8](#).

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.

For detailed configuration steps, see [Configuring Discard Extra Paths, on page 100](#).

Restrictions

These restrictions apply to the discard extra paths feature:

- 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 Default Limits

Cisco IOS XR 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

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 recurred 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.

For information on route policy filtering for next hops using the next-hop attach point, see the *Implementing Routing Policy Language on Cisco IOS XR Software* module of *Cisco IOS XR Routing Configuration Guide* (this publication).

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.

Scoped IPv4/VPNv4 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 and VPNv4 unicast address families share the same gateway context, because they are registered with the IPv4 unicast table in the RIB. As a result, both the global IPv4 unicast table and the VPNv4 table are processed when an IPv4 unicast next-hop notification is received from the RIB. A mask is maintained in the next hop, indicating if whether the next hop belongs to IPv4 unicast or VPNv4 unicast, or both. This scoped table walk localizes the processing in the appropriate address family table.

Reordered Address Family Processing

The Cisco IOS XR 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
- VPNv6 unicast
- IPv4 labeled unicast
- IPv4 unicast
- IPv4 MDT
- IPv4 multicast
- IPv6 unicast
- IPv6 multicast
- IPv6 labeled 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. See the *BGP Commands on Cisco IOS XR Software* module of *Routing Command Reference for Cisco CRS Routers* for information on the next-hop **show** and **clear** commands.

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. See the *BGP Debug Commands on Cisco IOS XR Software* module of *Cisco IOS XR Routing Debug Command Reference for the Cisco CRS-1 Router* .

Autonomous System Number Formats in BGP

Autonomous system numbers (ASNs) are globally unique identifiers used to identify autonomous systems (ASs) and enable ASs to exchange exterior routing information between neighboring ASs. A unique ASN is allocated to each AS for use in BGP routing. ASNs are encoded as 2-byte numbers and 4-byte numbers in BGP.

2-byte Autonomous System Number Format

The 2-byte ASNs are represented in asplain notation. The 2-byte range is 1 to 65535.

4-byte Autonomous System Number Format

To prepare for the eventual exhaustion of 2-byte Autonomous System Numbers (ASNs), BGP has the capability to support 4-byte ASNs. The 4-byte ASNs are represented both in asplain and asdot notations.

The byte range for 4-byte ASNs in asplain notation is 1-4294967295. The AS is represented as a 4-byte decimal number. The 4-byte ASN asplain representation is defined in [draft-ietf-idr-as-representation-01.txt](#).

For 4-byte ASNs in asdot format, the 4-byte range is 1.0 to 65535.65535 and the format is:

high-order-16-bit-value-in-decimal . low-order-16-bit-value-in-decimal

The BGP 4-byte ASN capability is used to propagate 4-byte-based AS path information across BGP speakers that do not support 4-byte AS numbers. See [draft-ietf-idr-as4bytes-12.txt](#) for information on increasing the size of an ASN from 2 bytes to 4 bytes. AS is represented as a 4-byte decimal number

as-format Command

The **as-format** command configures the ASN notation to asdot. The default value, if the **as-format** command is not configured, is asplain.

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:

```
RP/0/RP0/CPU0:router# configuration
RP/0/RP0/CPU0:router (config)# router bgp 140
RP/0/RP0/CPU0:router (config-bgp)#
```

Router Address Family Configuration Mode

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

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

Neighbor Configuration Mode

The following example shows how to enter neighbor configuration mode:

```
RP/0/RP0/CPU0:router (config)# router bgp 140
RP/0/RP0/CPU0:router (config-bgp)# neighbor 10.0.0.1
RP/0/RP0/CPU0:router (config-bgp-nbr)#
```

Neighbor Address Family Configuration Mode

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

```
RP/0/RP0/CPU0:router (config)# router bgp 112
RP/0/RP0/CPU0:router (config-bgp)# neighbor 10.0.0.1
RP/0/RP0/CPU0:router (config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router (config-bgp-nbr-af)#
```

VRF Configuration Mode

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

```
RP/0/RP0/CPU0:router (config)# router bgp 140
RP/0/RP0/CPU0:router (config-bgp)# vrf vrf_A
RP/0/RP0/CPU0:router (config-bgp-vrf)#
```

VRF Address Family Configuration Mode

The following example shows how to enter VRF 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)# address-family ipv4 unicast
RP/0/RP0/CPU0:router (config-bgp-vrf-af)#
```

Configuring Resilient Per-CE Label Mode Under VRF Address Family

Perform this task to configure resilient per-ce label mode under VRF address family.



Note Resilient per-CE 6PE label allocation is not supported on CRS-1 and CRS-3 routers, but supported only on ASR 9000 routers.

SUMMARY STEPS

1. **configure**
2. **router bgpas-number**
3. **vrfvrf-instance**
4. **address-family {ipv4 | ipv6} unicast**
5. **label mode per-ce**
6. Do one of the following:
 - **end**
 - **commit**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters global configuration mode.

Step 2 **router bgpas-number**

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 666
RP/0/RP0/CPU0:router(config-bgp)#
```

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

Step 3 **vrfvrf-instance**

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# vrf vrf-pe
RP/0/RP0/CPU0:router(config-bgp-vrf)#
```

Configures a VRF instance.

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

Example:

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

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

Step 5 label mode per-ce

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# label mode per-ce
RP/0/RP0/CPU0:router(config-bgp-vrf-af)#
```

Configures resilient per-ce label mode.

Step 6 Do one of the following:

- **end**
- **commit**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# end
```

or

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
 - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
 - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Resilient Per-CE Label Mode Using a Route-Policy

Perform this task to configure resilient per-ce label mode using a route-policy.



Note Resilient per-CE 6PE label allocation is not supported on CRS-1 and CRS-3 routers, but supported only on ASR 9000 routers.

SUMMARY STEPS

1. **configure**

2. `route-policy` *policy-name*
3. `set label mode per-ce`
4. Do one of the following:
 - `end`
 - `commit`

DETAILED STEPS

Step 1 `configure`

Example:

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

Enters global configuration mode.

Step 2 `route-policy` *policy-name*

Example:

```
RP/0/RP0/CPU0:router(config)# route-policy route1
RP/0/RP0/CPU0:router(config-rpl)#
```

Creates a route policy and enters route policy configuration mode.

Step 3 `set label mode per-ce`

Example:

```
RP/0/RP0/CPU0:router(config-rpl)# set label mode per-ce
RP/0/RP0/CPU0:router(config-rpl)#
```

Configures resilient per-ce label mode.

Step 4 Do one of the following:

- `end`
- `commit`

Example:

```
RP/0/RP0/CPU0:router(config-rpl)# end
```

or

```
RP/0/RP0/CPU0:router(config-rpl)# commit
```

Saves configuration changes.

- When you issue the `end` command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
```

- Entering `yes` saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

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)#
```

VPNv4 Address Family Configuration Mode

The following example shows how to enter VPNv4 address family configuration mode:

```
RP/0/RP0/CPU0:router(config)# router bgp 152
RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)#
```

VPNv6 Address Family Configuration Mode

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

```
RP/0/RP0/CPU0:router(config)# router bgp 150
RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv6 unicast
RP/0/RP0/CPU0:router(config-bgp-af)#
```

L2VPN Address Family Configuration Mode

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

```
RP/0/RP0/CPU0:router(config)# router bgp 100
RP/0/RP0/CPU0:router(config-bgp)# address-family l2vpn vpls-vpws
```

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.23.1.2
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002
RP/0/RP0/CPU0: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 Cisco IOS XR software, the configuration is as follows:

```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 2002::2
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2023
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# next-hop-self
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy one in
```

- You must enter neighbor-specific IPv4, IPv6, VPNv4, or VPNv6 commands in neighbor address-family configuration submode. In Cisco IOS XR software, the configuration is as follows:

```
RP/0/RP0/CPU0:router(config)# router bgp 109
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.40.24
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# maximum-prefix 1000
```

- You must enter neighbor-specific IPv4 and IPv6 commands in VRF neighbor address-family configuration submode. In Cisco IOS XR software, the configuration is as follows:

```
RP/0/RP0/CPU0:router(config)# router bgp 110
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)# route-policy pass all 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, VPNv4, or VPNv6 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 :

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# af-group afmcast1 address-family ipv4 unicast
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# session-group session1
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config)# router bgp 123
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group nbrgroup1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)#
```

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

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group nbrgroup1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.1.1.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# session-group AS_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 20
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.0.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbr)# use session-group AS_2
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```

RP/0/RP0/CPU0:router(config)# router bgp 150
RP/0/RP0/CPU0:router(config-bgp)# session-group AS_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 20
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.1.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0: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:

```

RP/0/RP0/CPU0: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.

```

RP/0/RP0/CPU0:routerconfig)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# session-group ADV
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group ADV_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 25
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# route-policy POLICY_1 in
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.2.2
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbr)# use session-group ADV

```

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group adv_15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# remote-as 10
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.101.5
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbr)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.101.10
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```

RP/0/RP0/CPU0:router(config)# router bgp 142
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# next-hop-self
RP/0/RP0/CPU0:router(config-bgp-afgrp)# route-policy POLICY_1 in
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# use session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# ebgp-multihop 3
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# weight 100
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# send-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit

RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.0.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# use af-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# weight 200

```

The following example displays sample output from the **show bgp neighbors** command using the **inheritance** keyword. The example shows that the neighbor inherits session parameters from neighbor group GROUP_1, which in turn inherits from session group GROUP_2. The neighbor inherits IPv4 unicast parameters from address family group GROUP_3 and IPv4 multicast parameters from neighbor group GROUP_1:

```
RP/0/RP0/CPU0:router# show bgp neighbors 192.168.0.1 inheritance

Session:          n:GROUP_1 s:GROUP_2
IPv4 Unicast:     a:GROUP_3
IPv4 Multicast:   n:GROUP_1
```

The following example displays sample output from the **show bgp neighbors** command using the **configuration** keyword. The example shows from where each item of configuration was inherited, or if it was configured directly on the neighbor (indicated by []). For example, the **ebgp-multihop 3** command was inherited from neighbor group GROUP_1 and the **next-hop-self** command was inherited from the address family group GROUP_3:

```
RP/0/RP0/CPU0:router# show bgp neighbors 192.168.0.1 configuration

neighbor 192.168.0.1
  remote-as 2                []
  advertisement-interval 15  [n:GROUP_1 s:GROUP_2]
  ebgp-multihop 3           [n:GROUP_1]
  address-family ipv4 unicast []
  next-hop-self             [a:GROUP_3]
  route-policy POLICY_1 in  [a:GROUP_3]
  weight 200                []
  address-family ipv4 multicast [n:GROUP_1]
  default-originate         [n:GROUP_1]
```

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:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# remove-private-as
RP/0/RP0/CPU0:router(config-bgp-afgrp)# route-policy POLICY_1 in
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_1 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-afgrp)# maximum-prefix 2500 75 warning-only
RP/0/RP0/CPU0:router(config-bgp-afgrp)# default-originate
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
```

```
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-extended-community-ebgp
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:


```
RP/0/RP0/CPU0:router(config)# router bgp 113
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-sngrp)# use session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# update-source Loopback 0
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# use session-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-sngrp)# ebgp-multihop 2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-sngrp)# dmz-link-bandwidth
```

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

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
```

```

RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# remove-private-as
RP/0/RP0/CPU0:router(config-bgp-afgrp)# soft-reconfiguration inbound
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-extended-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-afgrp)# capability orf prefix both
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-sngrp)# timers 30 90
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# remote-as 1982
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# use neighbor-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RP0/CPU0:router(config-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# use session-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# use af-group GROUP_2
RP/0/RP0/CPU0: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:

```

RP/0/RP0/CPU0: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:

```

RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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, starting from Cisco IOS XR Software Release 6.2.x, 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.

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:

```
RP/0/RP0/CPU0:router(config)# route-policy pass-all
RP/0/RP0/CPU0:router(config-rpl)# pass
RP/0/RP0/CPU0:router(config-rpl)# end-policy
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config)# router bgp 1
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.40.24
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 21
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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

Address Family: IPv4 Multicast
=====

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 1
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process          RecvTblVer    bRIB/RIB    SendTblVer
Speaker          1             1            1

Some configured eBGP neighbors do not have both inbound and
outbound policies configured for IPv4 Multicast address family.
These neighbors will default to sending and/or receiving no
routes and are marked with '!' in the output below. Use the
'show bgp neighbor <nbr_address>' command for details.

Neighbor         Spk   AS MsgRcvd MsgSent   TblVer  InQ  OutQ  Up/Down  St/PfxRcd
```

```

10.0.101.2      0      2      0      0      0      0      0 00:00:00 Idle!

Address Family: IPv6 Unicast
=====

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 2
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process          RecvTblVer    bRIB/RIB    SendTblVer
Speaker          2             2             2

Neighbor        Spk    AS  MsgRcvd  MsgSent    TblVer  InQ  OutQ  Up/Down  St/PfxRcd
2222::2         0      2    920     918         2       0    0  15:15:11  1
2222::4         0      3     0       0         0       0    0  00:00:00 Idle!

Address Family: IPv6 Multicast
=====

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 1
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process          RecvTblVer    bRIB/RIB    SendTblVer
Speaker          1             1             1

Some configured eBGP neighbors do not have both inbound and
outbound policies configured for IPv6 Multicast address family.
These neighbors will default to sending and/or receiving no
routes and are marked with '!' in the output below. Use the
'show bgp neighbor <nbr_address>' command for details.

Neighbor        Spk    AS  MsgRcvd  MsgSent    TblVer  InQ  OutQ  Up/Down  St/PfxRcd
2222::2         0      2    920     918         0       0    0  15:15:11  0
2222::4         0      3     0       0         0       0    0  00:00:00 Idle!

```

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. See the *Implementing Routing Policy on Cisco IOS XR Software* module in the *Routing Configuration Guide for Cisco CRS Routers* for details on table policy use. See the *Cisco Express Forwarding Commands on Cisco IOS XR Software* module in the *IP Addresses and Services Command Reference for Cisco CRS Routers* for details on BGP policy accounting.

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.

Update Groups

The BGP Update Groups feature contains an algorithm that dynamically calculates and optimizes update groups of neighbors that share outbound policies and can share the update messages. The BGP Update Groups feature separates update group replication from peer group configuration, improving convergence time and flexibility of neighbor configuration.

To use this feature, you must understand the following concepts:

Related Topics

[BGP Update Generation and Update Groups](#) , on page 32

[BGP Update Group](#) , on page 32

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 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 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.

The cost community attribute is applied to internal routes by configuring the **set extcommunity cost** command in a route policy. See the *Routing Policy Language Commands on Cisco IOS XR Software* module of *Cisco IOS XR Routing Command Reference* for information on the **set extcommunity cost** command. The cost community set clause is configured with a cost community ID number (0–255) and cost community number (0–4294967295). The cost community number determines the preference for the path. The path with the lowest cost community number is preferred. Paths that are not specifically configured with the cost community number are assigned a default cost community number of 2147483647 (the midpoint between 0 and 4294967295) and evaluated by the best-path selection process accordingly. When two paths have been configured with the same cost community number, the path selection process prefers the path with the lowest cost community ID. The cost-extended community attribute is propagated to iBGP peers when extended community exchange is enabled.

The following commands include the **route-policy** keyword, which you can use to apply a route policy that is configured with the cost community set clause:

- **aggregate-address**

- redistribute
- network

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. See the scenario described in [Influencing Route Preference in a Multiexit IGP Network, on page 35](#).



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

See [BGP Best Path Algorithm, on page 37](#) for information on the BGP best-path selection process.

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

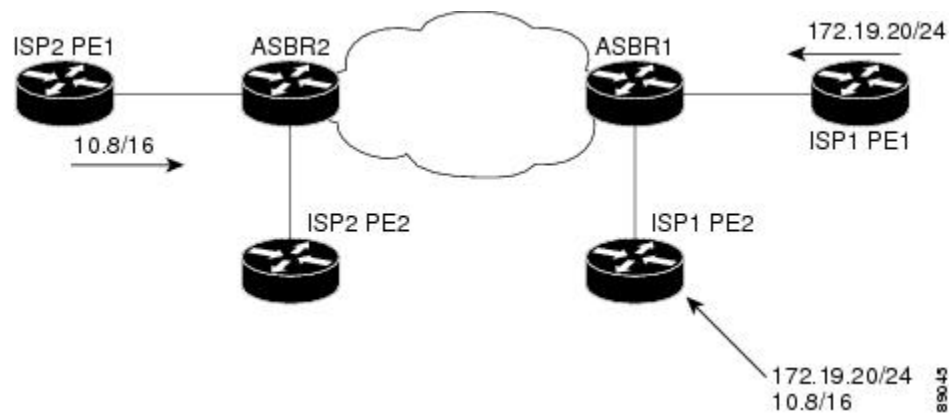
The single advertised path includes the aggregate cost communities as follows:

{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 1: 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:

```
RP/0/RP0/CPU0:router(config)# route-policy ISP2_PE1
RP/0/RP0/CPU0: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.

BGP Cost Community Support for EIGRP MPLS VPN PE-CE with Back-door Links

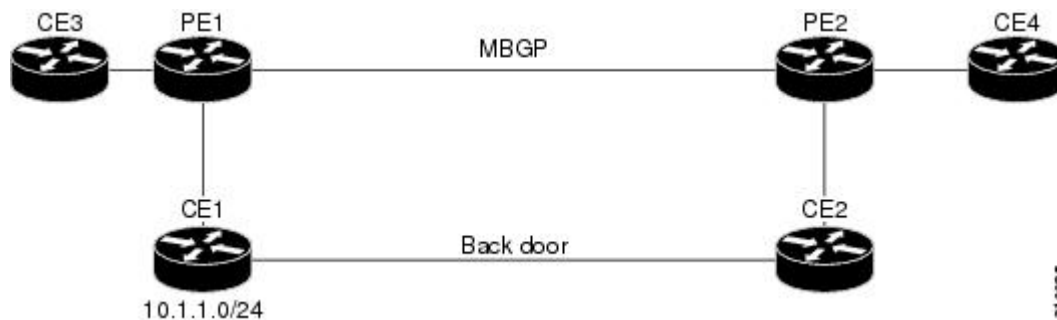
Back-door links in an EIGRP MPLS VPN topology is preferred by BGP if the back-door link is learned first. (A back-door link, or route, is a connection that is configured outside of the VPN between a remote and main site; for example, a WAN leased line that connects a remote site to the corporate network.)

The “prebest path” point of insertion (POI) in the BGP cost community feature supports mixed EIGRP VPN network topologies that contain VPN and back-door links. This POI is applied automatically to EIGRP routes that are redistributed into BGP. The “prebest path” POI carries the EIGRP route type and metric. This POI influences the best-path calculation process by influencing BGP to consider the POI before any other comparison step. No configuration is required. This feature is enabled automatically for EIGRP VPN sites when Cisco IOS XR software is installed on a PE, CE, or back-door router.

For information about configuring EIGRP MPLS VPNs, see the *MPLS Configuration Guide for the Cisco CRS Routers*.

Figure 2: Network Showing How Cost Community Can be Used to Support Backdoor Links

This figure shows how cost community can be used to support backdoor links in a network.



The following sequence of events happens in PE1:

1. PE1 learns IPv4 prefix 10.1.1.0/24 from CE1 through EIGRP running a virtual routing and forwarding (VRF) instance. EIGRP selects and installs the best path in the RIB. It also encodes the cost-extended community and adds the information to the RIB.
2. The route is redistributed into BGP (assuming that IGP-to-BGP redistribution is configured). BGP also receives the cost-extended community from the route through the redistribution process.
3. After BGP has determined the best path for the newly redistributed prefix, the path is advertised to PE peers (PE2).
4. PE2 receives the BGP VPNv4 prefix route `_distinguisher:10.1.1.0/24` along with the cost community. It is likely that CE2 advertises the same prefix (because of the back-door link between CE1 and CE2) to PE2 through EIGRP. PE2 BGP would have already learned the CE route through the redistribution process along with the cost community value.
5. PE2 has two paths within BGP: one with cost community `cost1` through multipath BGP (PE1) and another with cost community `cost2` through the EIGRP neighbor (CE2).
6. PE2 runs the enhanced BGP best-path calculation.
7. PE2 installs the best path in the RIB passing the appropriate cost community value.
8. PE2 RIB has two paths for 10.1.1.0/24: one with cost community `cost2` added by EIGRP and another with the cost community `cost1` added by BGP. Because both the route paths have cost community, RIB

compares the costs first. The BGP path has the lower cost community, so it is selected and downloaded to the RIB.

9. PE2 RIB redistributes the BGP path into EIGRP with VRF. EIGRP runs a diffusing update algorithm (DUAL) because there are two paths, and selects the BGP-redistributed path.
10. PE2 EIGRP advertises the path to CE2 making the path the next hop for the prefix to send the traffic over the MPLS network.

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. See [BGP Best Path Algorithm, on page 37](#) for information on the BGP best-path algorithm.

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 eIBGP specifies internal and external BGP multipath peers. eIBGP 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 [#unique_72](#) 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 [#unique_72](#).
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 [#unique_72](#).
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 [#unique_72](#).
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 [#unique_72](#), 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. For information on specifying the administrative distance for BGP, see the BGP Commands module of the *Routing Command Reference for Cisco CRS Routers*

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 [Table 3: BGP Default Administrative Distances, on page 41](#).

Table 3: BGP Default Administrative Distances

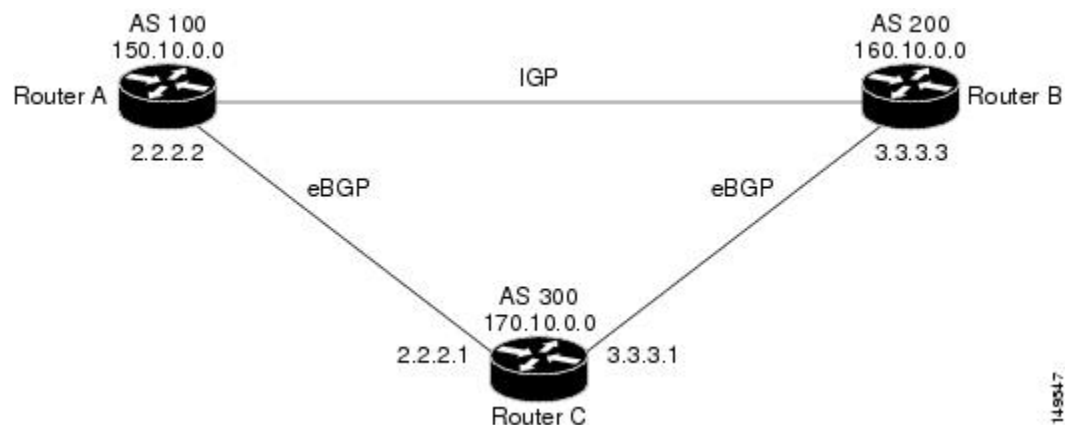
Distance	Default Value	Function
External	20	Applied to routes learned from eBGP.
Internal	200	Applied to routes learned from iBGP.
Local	200	Applied to routes originated by the router.



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. See [Figure 3: Back Door Example , on page 41](#).

Figure 3: Back Door Example



In [Figure 3: Back Door Example , on page 41](#), 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:

```
RP/0/RP0/CPU0:router(config)# router bgp 100
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0: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.

Multiprotocol BGP

Multiprotocol BGP is an enhanced BGP that carries routing information for multiple network layer protocols and IP multicast routes. BGP carries two sets of routes, one set for unicast routing and one set for multicast routing. The routes associated with multicast routing are used by the Protocol Independent Multicast (PIM) feature to build data distribution trees.

Multiprotocol BGP is useful when you want a link dedicated to multicast traffic, perhaps to limit which resources are used for which traffic. Multiprotocol BGP allows you to have a unicast routing topology different from a multicast routing topology providing more control over your network and resources.

In BGP, the only way to perform interdomain multicast routing was to use the BGP infrastructure that was in place for unicast routing. Perhaps you want all multicast traffic exchanged at one network access point (NAP). If those routers were not multicast capable, or there were differing policies for which you wanted multicast traffic to flow, multicast routing could not be supported without multiprotocol BGP.



Note

It is possible to configure BGP peers that exchange both unicast and multicast network layer reachability information (NLRI), but you cannot connect multiprotocol BGP clouds with a BGP cloud. That is, you cannot redistribute multiprotocol BGP routes into BGP.

[Figure 4: Noncongruent Unicast and Multicast Routes, on page 43](#) illustrates simple unicast and multicast topologies that are incongruent, and therefore are not possible without multiprotocol BGP.

Autonomous systems 100, 200, and 300 are each connected to two NAPs that are FDDI rings. One is used for unicast peering (and therefore the exchange of unicast traffic). The Multicast Friendly Interconnect (MFI) ring is used for multicast peering (and therefore the exchange of multicast traffic). Each router is unicast and multicast capable.

Figure 4: Noncongruent Unicast and Multicast Routes

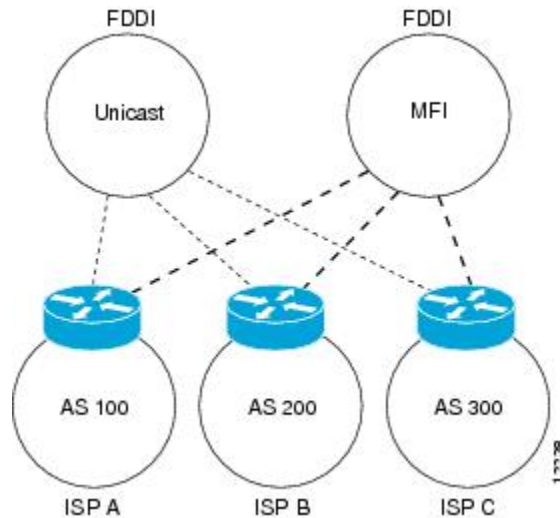


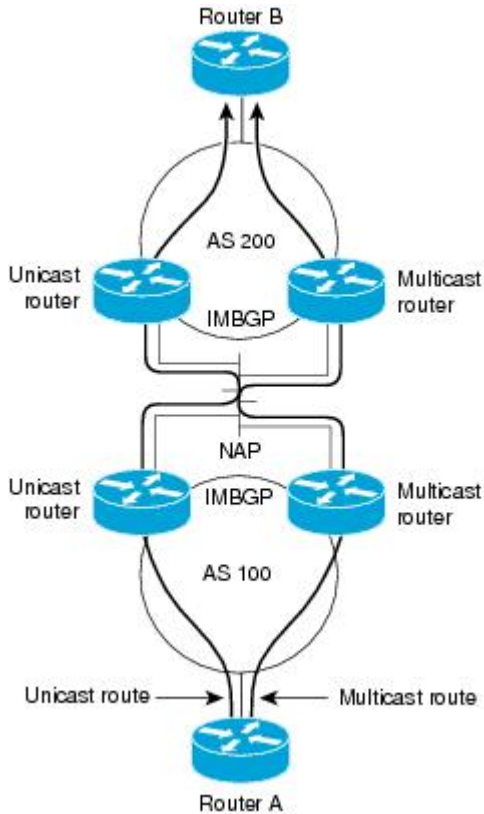
Figure 5: Multicast BGP Environment, on page 44 is a topology of unicast-only routers and multicast-only routers. The two routers on the left are unicast-only routers (that is, they do not support or are not configured to perform multicast routing). The two routers on the right are multicast-only routers. Routers A and B support both unicast and multicast routing. The unicast-only and multicast-only routers are connected to a single NAP.

In Figure 5: Multicast BGP Environment, on page 44, only unicast traffic can travel from Router A to the unicast routers to Router B and back. Multicast traffic could not flow on that path, so another routing table is required. Multicast traffic uses the path from Router A to the multicast routers to Router B and back.

Figure 5: Multicast BGP Environment, on page 44 illustrates a multiprotocol BGP environment with a separate unicast route and multicast route from Router A to Router B. Multiprotocol BGP allows these routes to be incongruent. Both of the autonomous systems must be configured for internal multiprotocol BGP (IMBGP) in the figure.

A multicast routing protocol, such as PIM, uses the multicast BGP database to perform Reverse Path Forwarding (RPF) lookups for multicast-capable sources. Thus, packets can be sent and accepted on the multicast topology but not on the unicast topology.

Figure 5: Multicast BGP Environment



11754

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.

Minimizing 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

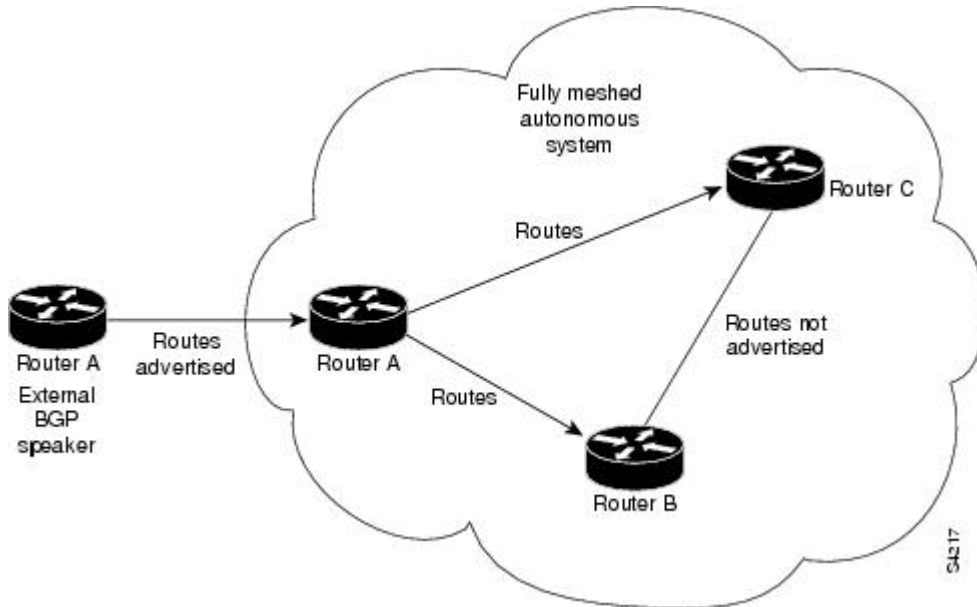
One way to reduce the iBGP mesh is to divide an autonomous system into multiple subautonomous systems and group them into a single confederation. To the outside world, the confederation looks like a single autonomous system. Each autonomous system is fully meshed within itself and has a few connections 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 Route Reflectors

BGP requires that all iBGP speakers be fully meshed. However, this requirement does not scale well when there are many iBGP speakers. Instead of configuring a confederation, you can reduce the iBGP mesh by using a route reflector configuration.

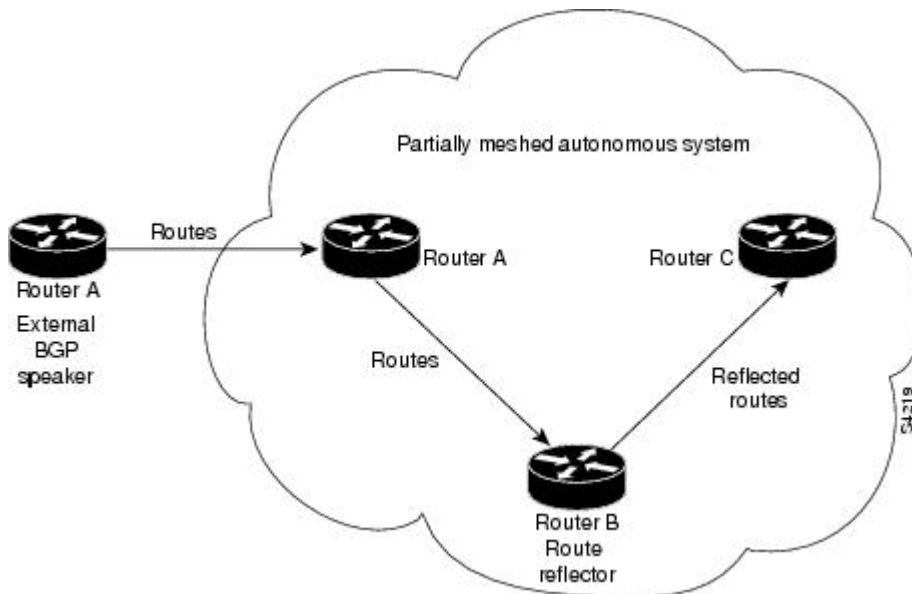
[Figure 6: Three Fully Meshed iBGP Speakers, on page 46](#) illustrates a simple iBGP configuration with three iBGP speakers (routers A, B, and C). Without route reflectors, when Router A receives a route from an external neighbor, it must advertise it to both routers B and C. Routers B and C do not readvertise the iBGP learned route to other iBGP speakers because the routers do not pass on routes learned from internal neighbors to other internal neighbors, thus preventing a routing information loop.

Figure 6: Three Fully Meshed iBGP Speakers



With route reflectors, all iBGP speakers need not be fully meshed because there is a method to pass learned routes to neighbors. In this model, an iBGP peer is configured to be a route reflector responsible for passing iBGP learned routes to a set of iBGP neighbors. In [Figure 7: Simple BGP Model with a Route Reflector, on page 46](#), Router B is configured as a route reflector. When the route reflector receives routes advertised from Router A, it advertises them to Router C, and vice versa. This scheme eliminates the need for the iBGP session between routers A and C.

Figure 7: Simple BGP Model with a Route Reflector



The internal peers of the route reflector are divided into two groups: client peers and all other routers in the autonomous system (nonclient peers). A route reflector reflects routes between these two groups. The route reflector and its client peers form a *cluster*. The nonclient peers must be fully meshed with each other, but the

client peers need not be fully meshed. The clients in the cluster do not communicate with iBGP speakers outside their cluster.

Figure 8: More Complex BGP Route Reflector Model

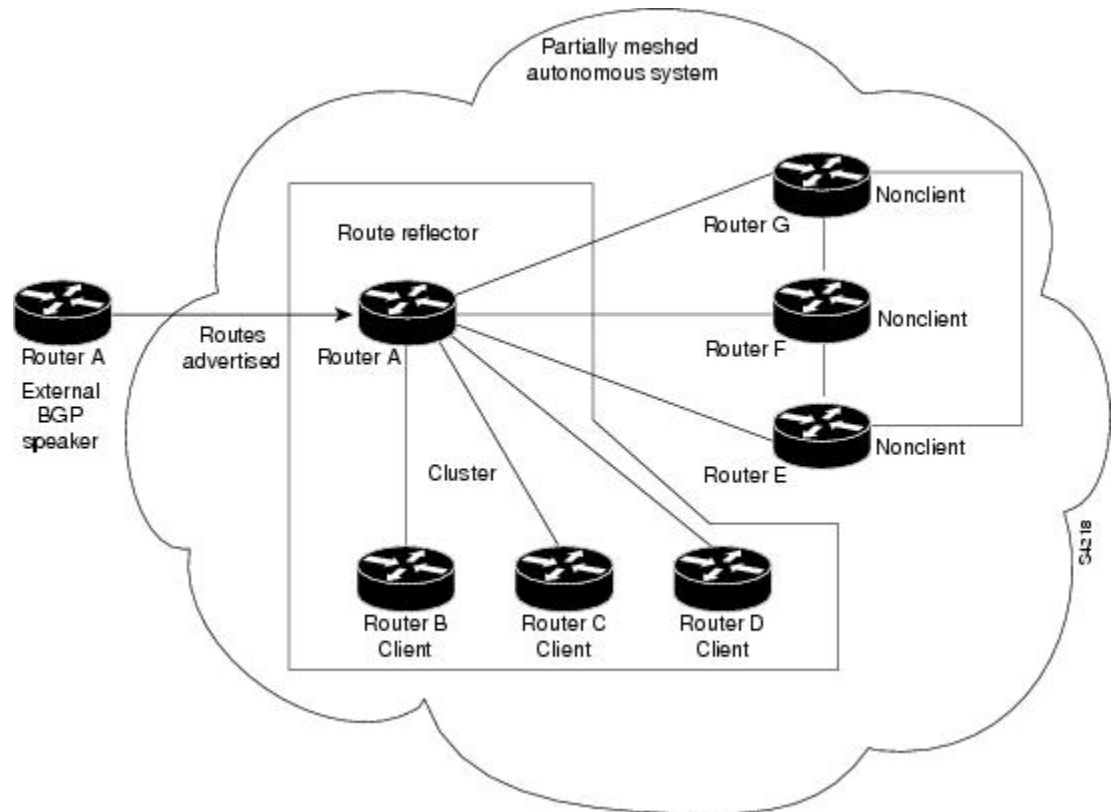


Figure 8: More Complex BGP Route Reflector Model, on page 47 illustrates a more complex route reflector scheme. Router A is the route reflector in a cluster with routers B, C, and D. Routers E, F, and G are fully meshed, nonclient routers.

When the route reflector receives an advertised route, depending on the neighbor, it takes the following actions:

- A route from an external BGP speaker is advertised to all clients and nonclient peers.
- A route from a nonclient peer is advertised to all clients.
- A route from a client is advertised to all clients and nonclient peers. Hence, the clients need not be fully meshed.

Along with route reflector-aware BGP speakers, it is possible to have BGP speakers that do not understand the concept of route reflectors. They can be members of either client or nonclient groups, allowing an easy and gradual migration from the old BGP model to the route reflector model. Initially, you could create a single cluster with a route reflector and a few clients. All other iBGP speakers could be nonclient peers to the route reflector and then more clusters could be created gradually.

An autonomous system can have multiple route reflectors. A route reflector treats other route reflectors just like other iBGP speakers. A route reflector can be configured to have other route reflectors in a client group or nonclient group. In a simple configuration, the backbone could be divided into many clusters. Each route

reflector would be configured with other route reflectors as nonclient peers (thus, all route reflectors are fully meshed). The clients are configured to maintain iBGP sessions with only the route reflector in their cluster.

Usually, a cluster of clients has a single route reflector. In that case, the cluster is identified by the router ID of the route reflector. To increase redundancy and avoid a single point of failure, a cluster might have more than one route reflector. In this case, all route reflectors in the cluster must be configured with the cluster ID so that a route reflector can recognize updates from route reflectors in the same cluster. All route reflectors serving a cluster should be fully meshed and all of them should have identical sets of client and nonclient peers.

By default, the clients of a route reflector are not required to be fully meshed and the routes from a client are reflected to other clients. However, if the clients are fully meshed, the route reflector need not reflect routes to clients.

As the iBGP learned routes are reflected, routing information may loop. The route reflector model has the following mechanisms to avoid routing loops:

- Originator ID is an optional, nontransitive BGP attribute. It is a 4-byte attributed created by a route reflector. The attribute carries the router ID of the originator of the route in the local autonomous system. Therefore, if a misconfiguration causes routing information to come back to the originator, the information is ignored.
- Cluster-list is an optional, nontransitive BGP attribute. It is a sequence of cluster IDs that the route has passed. When a route reflector reflects a route from its clients to nonclient peers, and vice versa, it appends the local cluster ID to the cluster-list. If the cluster-list is empty, a new cluster-list is created. Using this attribute, a route reflector can identify if routing information is looped back to the same cluster due to misconfiguration. If the local cluster ID is found in the cluster-list, the advertisement is ignored.

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 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.

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:

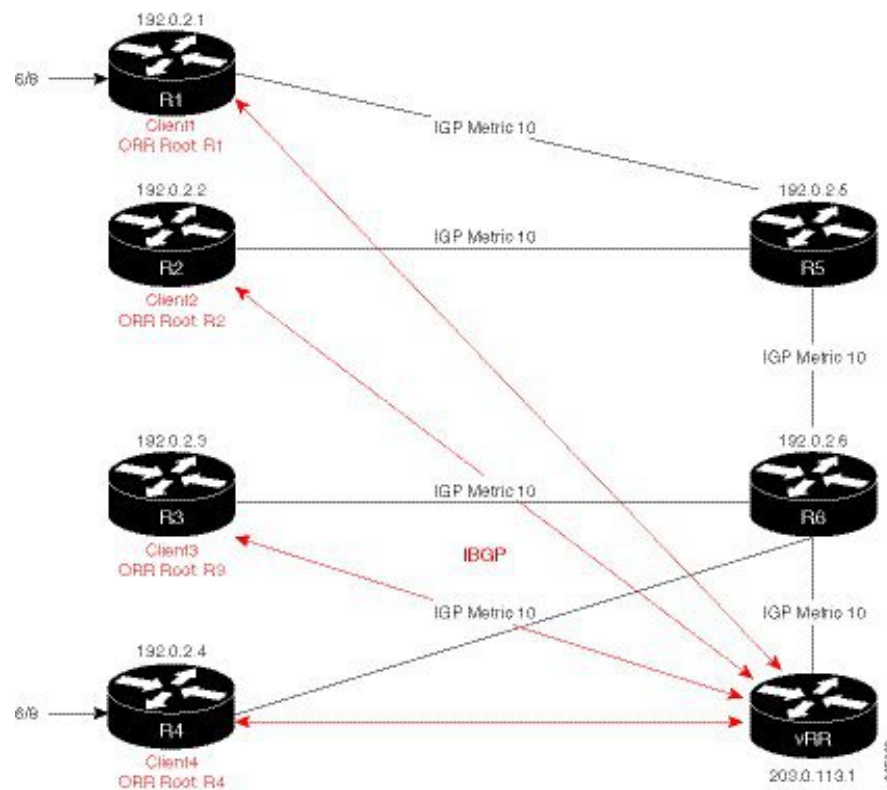
- calculates the bestpath from the point of view of a RR client.
- enables vRR to be placed anywhere in the topology or in a SP data center.
- allows SPs to scale memory and CPU usage of RR deployments.

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 9: 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
```

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
```

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 path 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 blacklist filtering

Configuring 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.

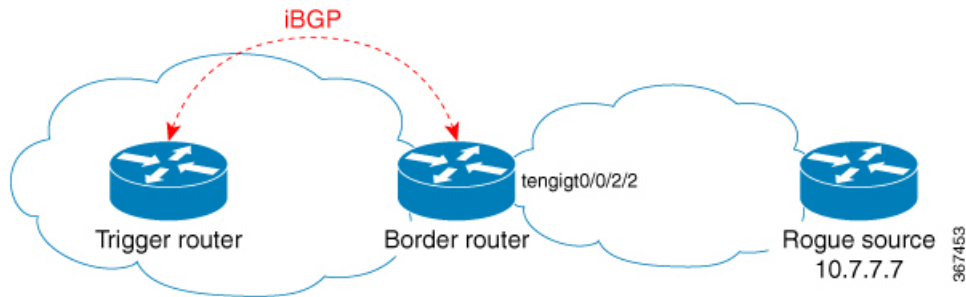
RTBH filtering sets the next-hop of the victim's prefix to the null interface. The traffic destined to the victim is dropped at the ingress.

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 10: 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

```

Verification

On the border router, the prefix 10.7.7.7/32 is flagged as Nexthop-discard:

```

RP/0/RSP0/CPU0:router#show bgp
BGP router identifier 10.210.0.5, local AS number 65001
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0000000 RD version: 12
BGP main routing table version 12
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
N>i10.7.7.7/32	192.168.102.2	0	100	0	?

```

RP/0/RSP0/CPU0:router#show bgp 10.7.7.7/32
BGP routing table entry for 10.7.7.7/32
Versions:
  Process          bRIB/RIB  SendTblVer
  Speaker          12        12
Last Modified: Jul  4 14:37:29.048 for 00:20:52
Paths: (1 available, best #1, not advertised to EBGp peer)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    192.168.102.2 (discarded) from 192.168.102.2 (10.210.0.2)
      Origin incomplete, metric 0, localpref 100, valid, internal best, group-best
      Received Path ID 0, Local Path ID 1, version 12
      Community: 1234:4321 no-export

RP/0/RSP0/CPU0:router#show route 10.7.7.7/32

Routing entry for 10.7.7.7/32
  Known via "bgp 65001", distance 200, metric 0, type internal
  Installed Jul  4 14:37:29.394 for 01:47:02
  Routing Descriptor Blocks
    directly connected, via Null0
      Route metric is 0
  No advertising protos.

```

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 [Configuring Per Neighbor TCP MSS, on page 101](#).

For detailed steps to disable per neighbor TCP MSS, see [Disabling Per Neighbor TCP MSS, on page 103](#).

MPLS VPN Carrier Supporting Carrier

Carrier supporting carrier (CSC) is a term used to describe a situation in which one service provider allows another service provider to use a segment of its backbone network. The service provider that provides the

segment of the backbone network to the other provider is called the *backbone carrier*. The service provider that uses the segment of the backbone network is called the *customer carrier*.

A backbone carrier offers Border Gateway Protocol and Multiprotocol Label Switching (BGP/MPLS) VPN services. The customer carrier can be either:

- An Internet service provider (ISP) (By definition, an ISP does not provide VPN service.)
- A BGP/MPLS VPN service provider

You can configure a CSC network to enable BGP to transport routes and MPLS labels between the backbone carrier provider edge (PE) routers and the customer carrier customer edge (CE) routers using multiple paths. The benefits of using BGP to distribute IPv4 routes and MPLS label routes are:

- BGP takes the place of an Interior Gateway Protocol (IGP) and Label Distribution Protocol (LDP) in a VPN routing and forwarding (VRF) table. You can use BGP to distribute routes and MPLS labels. Using a single protocol instead of two simplifies the configuration and troubleshooting.
- BGP is the preferred routing protocol for connecting two ISPs, mainly because of its routing policies and ability to scale. ISPs commonly use BGP between two providers. This feature enables those ISPs to use BGP.

For detailed information on configuring MPLS VPN CSC with BGP, see the *Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software* module of the *MPLS Configuration Guide for the Cisco CRS Routers*.

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.

See the *System Security Configuration Guide for Cisco CRS Routers* for information on keychain management.

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 Multicast VPN

The BGP Multicast VPN feature uses the IPv4 multicast distribution tree (MDT) subaddress family identifier (SAFI) in Border Gateway Protocol (BGP).

Multicast VPN (MVPN) extends the VPN architecture to provide multicast services over a shared service provider backbone using native multicast technology. This is achieved using virtual connections between provider edge (PE) routers in each VPN and using native multicast forwarding inside the provider network. An MDT may span across multiple customer sites and the provider network, allowing traffic to flow freely from one source to multiple receivers.

MVPN is supported on VPN networks based on MPLS and on networks based on IP Layer 2 Tunnel Protocol version 3 (L2TPv3).

PE routers are the only routers that must be MVPN-aware and that must be able to signal to remote PEs information regarding the MVPN. Therefore, all PE routers must have a BGP relationship with each other—either directly or using a route reflector (RR).

Generally the source address of the default MDT is the same address used to source the internal BGP (iBGP) sessions with the remote PE routers that belong to the same VPN and multicast VPN routing and forwarding (MVRF) instance. When Protocol Independent Multicast–Source Specific Multicast (PIM–SSM) is used for transport inside the provider core, it is through the BGP relationship that the PEs indicate that they are MVPN-capable and provide for source discovery. This capability is indicated using the updated BGP message.



Note The source address can also be configured uniquely per VRF instance under multicast-routing configuration. See *Multicast Configuration Guide for Cisco CRS Routers*.

When a PE receives a BGP update, which includes the rendezvous point (RP) and the group information, it joins the root of that tree, thereby joining the MDT.

Figure 11: Multiprotocol iBGP Updates for MVPN, on page 59 shows Multiprotocol iBGP updates for MVPN. On PE1, PE2 is configured as its iBGP peer. This BGP peer configuration within a VRF triggers the MP-iBGP updates that send PE1 local VPN routes to PE2. BGP process on PE2 receives the VPN updates and installs VPN routes in the Routing Information Base (RIB) VRF table. When PIM looks up a VRF source or rendezvous point address that is reachable through the provider core, it receives an MP-iBGP route from the RIB.

When an MVPN-specific default MDT group is configured on PE1, PIM creates a virtual MDT tunnel interface with the tunnel source address the same as the BGP local peering address. This MDT interface is used by PIM to send VPN packets to the provider network and to receive VPN packets from the provider network. PIM also exchanges control messages over this MDT interface.

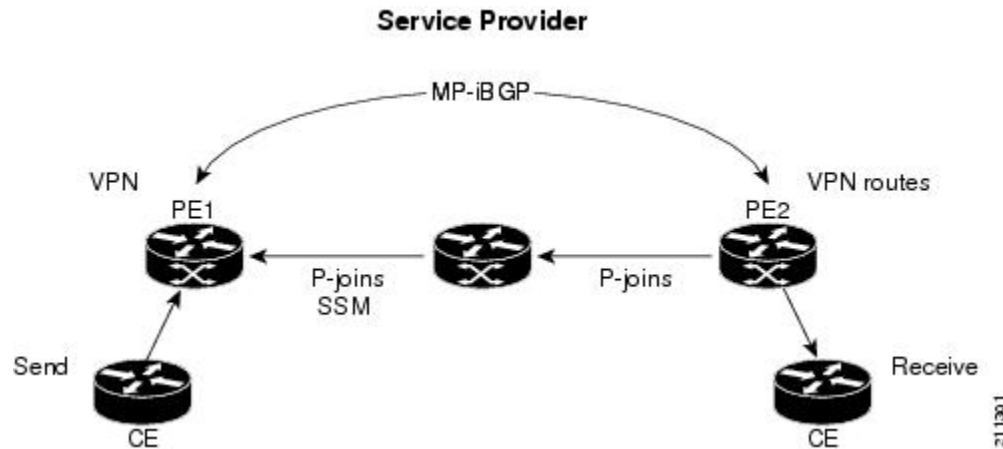
Each time a default MDT group is configured for a specific VRF, BGP builds an MDT SAFI update, with network layer reachability information (NLRI) containing the local PE BGP peering address and the newly configured MDT group address (The NLRI format is 8-byte-RD:IPv4-address followed by the MDT group address). This update is sent to all the BGP peers including PE2. The BGP process on PE2 receives this MDT update and notifies PIM. If the group is a PIM–SSM group, PIM on PE2 begins sending SSM joins to the BGP peering address on PE1 to establish an SSM tree in the core. This SSM tree is used to carry PIM control traffic and multicast data traffic in the corresponding VRF.

In summary, PIM requires the following from BGP:

- A new BGP MDT SAFI, which carries the VRF RD and BGP local peering address and default MDT group in its NLRI.
- A notification mechanism from BGP to PIM about the availability of the MDT SAFI update.
- A notification mechanism from PIM to BGP about the default MDT group address and source address.

See Internet Engineering Task Force (IETF) draft-nalawade-idr-mdt-safi-03 for detailed information on MDT SAFI.

Figure 11: Multiprotocol iBGP Updates for MVPN



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. commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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 4	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.
Step 5	address-family { vpn4 vpn6 } unicast Example: RP/0/RP0/CPU0:router(config-bgp)# address-family vpn4 unicast	Specifies the address family and enters the address family configuration submode. 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 vpn6 keyword
Step 6	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.
Step 7	address-family ipv4 mdt Example:	Specifies the multicast distribution tree (MDT) address family.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router (config-bgp) # address-family ipv4 mdt	
Step 8	exit Example: RP/0/RP0/CPU0:router (config-bgp-af) # exit	Exits the current configuration mode.
Step 9	neighbor <i>ip-address</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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 { <i>ipv4</i> <i>ipv6</i> } unicast Example: RP/0/RP0/CPU0: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 13	exit Example: RP/0/RP0/CPU0:router (config-bgp-nbr-af) # exit	(Optional) Exits the current configuration mode.
Step 14	address-family { <i>vpn4</i> <i>vpn6</i> } unicast Example: RP/0/RP0/CPU0:router (config-bgp-nbr) # address-family vpn4 unicast	(Optional) Enters address family configuration submenu for the specified address family. Note Required if you are configuring multicast MVPN. If configuring MVPNv6, use the vpn6 keyword.
Step 15	exit Example:	Exits the current configuration mode.

	Command or Action	Purpose
	<pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# exit</pre>	
Step 16	address-family ipv4 mdt Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 mdt</pre>	Specifies the multicast distribution tree (MDT) address family.
Step 17	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-af)# exit</pre>	Exits the current configuration mode.
Step 18	vrf vrf-name Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# vrf vpn1</pre>	(Optional) Enables BGP routing for a particular VRF on the PE router. Note Required if you are configuring multicast MVPN.
Step 19	rd { as-number:nn ip-address:nn auto } Example: <pre>RP/0/RP0/CPU0:router(config-bgp-vrf)# rd 1:1</pre>	(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: <pre>RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast</pre>	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 metric-value] [route-policy route-policy-name] 	(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 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>] • redistribute static [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf-af) # redistribute eigrp 23</pre>	
Step 22	commit	

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 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.

- In-Service System Upgrade (ISSU)
- Minimum Disruption Restart (MDR)

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

NSR can be provisioned on a multishelf router. The following guidelines should be observed when provisioning NSR on a multishelf router:

- When provisioning NSR for line cards installed on a single rack, provision the active and standby applications on the distributed route processor (DRP) of that rack. If a rack failure occurs, sessions are dropped, because all line cards go down.

- When provisioning NSR for line cards installed on different racks, use one of the following three options:
 - Provision the active and standby applications on a distributed route processor (DRP) redundant pair, where there is a separate route processor in each rack. This configuration uses up two revenue-producing line-card slots on each rack, but is the most secure configuration.
 - Provision the active and standby applications on a distributed route processor (DRP) pair that spans two racks. In this configuration, the active/standby role of the line cards is not dependent on the active/standby role of the DRPs. This is called *flexible process redundancy* and provides for rack loss and efficient use of LC slots. Use of distributed BGP is not required with this solution.



Note Sessions on line cards in a lost rack are not protected with any of the above options, because there is no line-card redundancy. These options ensure only that sessions on other racks are not affected by a lost rack. However, lost sessions from a lost rack may cause some traffic loss on other racks, because destinations learned through those lost sessions may no longer have alternate routes. Also, rack loss may cause the CPUs on route processors of active racks to slow as they attempt to define new paths for some routes.

BGP Best-External Path

The Border Gateway Protocol (BGP) 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

The **advertise best-external** command enables the advertisement of the best-external path in global address family configuration mode and VRF address family configuration mode.

BGP Local Label Retention

When a primary PE-CE link fails, BGP withdraws the route corresponding to the primary path along with its local label and programs the backup path in the Routing Information Base (RIB) and the Forwarding Information Base (FIB), by default.

However, until all the internal peers of the primary PE reconverge to use the backup path as the new bestpath, the traffic continues to be forwarded to the primary PE with the local label that was allocated for the primary path. Hence the previously allocated local label for the primary path must be retained on the primary PE for some configurable time after the reconvergence. BGP Local Label Retention feature enables the retention of the local label for a specified period. If no time is specified, the local label is retained for a default value of five minutes.

The **retain local-label** command enables the retention of the local label until the network is converged.

BGP Over GRE Interfaces

Cisco IOS XR software provides the capability to run Border Gateway Protocol (BGP) over Generic Routing Encapsulation (GRE) tunnel interfaces.

GRE protocol transports packets of one protocol over another protocol by means of encapsulation. Service Providers can provide IP, MPLS VPN or L2VPN services between their networks that are connected together by a public network using GRE encapsulation to carry data securely over the public network.

The packet that needs to be transported is first encapsulated in a GRE header, which is further encapsulated in another protocol like IPv4 or IPv6 and then forwarded to the destination.

The Cisco IOS XR software GRE implementation is compliant with GRE encapsulation defined in RFC 2784. Key and Sequence numbering as defined in RFC 2890 is not supported in Cisco IOS XR software GRE implementation. To be backward compliant with RFC 1701, Cisco IOS XR software transmits GRE packets with Reserved0 field set to zero. A receiver that is compliant with RFC 1701 treats key present, sequence number, and strict source route as zero and do not expect key and sequence number. The Cisco IOS XR software discards a GRE packet with any of the bits in Reserved0 field set.

Command Line Interface (CLI) Consistency for BGP Commands

From Cisco IOS XR Release 3.9.0 onwards, 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

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.



Note BGP Additional Path feature is not supported under vrf.

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.



Note Add path is not be supported with MDT, tunnel, and L2VPN address families and eBGP peerings.

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.

When there are multiple border BGP routers having reachability information heard over eBGP, if no local policy is applied, the border routers will choose their eBGP paths as best. They advertise that bestpath inside the ISP network. For a core router, there can be multiple paths to the same destination, but it will select only one path as best and use that path for forwarding. iBGP multipath load sharing adds the ability to enable load sharing among multiple equi-distant paths.

Configuring multiple iBGP best paths enables a router to evenly share the traffic destined for a particular site.

The iBGP Multipath Load Sharing feature functions similarly in a Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) with a service provider backbone.

For multiple paths to the same destination to be considered as multipaths, the following criteria must be met:

- All attributes must be the same. The attributes include weight, local preference, autonomous system path (entire attribute and not just length), origin code, Multi Exit Discriminator (MED), and Interior Gateway Protocol (iGP) distance.
- The next hop router for each multipath must be different.

Even if the criteria are met and multiple paths are considered multipaths, the BGP speaking router will still designate one of the multipaths as the best path and advertise this best path to its neighbors.

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 eIBGP multipath as that might cause loops. Only per prefix label supports per VRF label mode.

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.

The following behavior is to be noted while using BGP selective multipath:

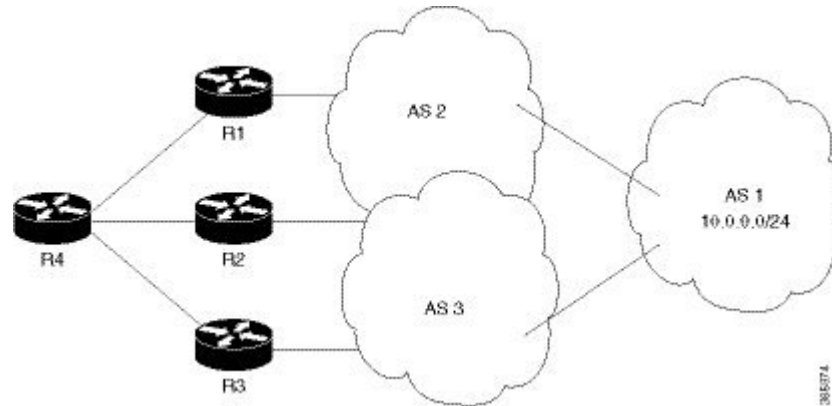
- BGP selective multipath does not impact best path calculations. A best path is always included in the set of multipaths.
- For VPN prefixes, the PE paths are *always* eligible to be multipaths.

For information on the **maximum-paths** and **multipath** commands, see the *Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference*.

Topology

A sample topology to illustrate the configuration used in this section is shown in the following figure.

Figure 12: 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
RP/0/RP0/CPU0:router(config)# router bgp 1
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)# maximum-paths ibgp 4 selective
RP/0/RP0/CPU0:router(config-bgp-af)# maximum-paths ebgp 5 selective
RP/0/RP0/CPU0:router(config-bgp-af)# commit

/* To configure selective multipath for eiBGP
RP/0/RP0/CPU0:router(config)# router bgp 1
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)# maximum-paths eiBGP 6 selective
RP/0/RP0/CPU0: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.

```

RP/0/RP0/CPU0:router(config-bgp)# neighbor 1.1.1.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# multipath
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit

RP/0/RP0/CPU0:router(config-bgp-nbr)# neighbor 2.2.2.2
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# multipath
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit

RP/0/RP0/CPU0:router(config-bgp-nbr)# neighbor 3.3.3.3
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit

```

You have successfully configured the BGP selective multipath feature.

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.

Per VRF and Per CE Label for IPv6 Provider Edge

The per VRF and per CE label for IPv6 feature makes it possible to save label space by allocating labels per default VRF or per CE nexthop.

All IPv6 Provider Edge (6PE) labels are allocated per prefix by default. Each prefix that belongs to a VRF instance is advertised with a single label, causing an additional lookup to be performed in the VRF forwarding table to determine the customer edge (CE) next hop for the packet.

However, use the **label mode** command with the **per-ce** keyword or the **per-vrf** keyword to avoid the additional lookup on the PE router and conserve label space.

Use **per-ce** keyword to specify that the same label be used for all the routes advertised from a unique customer edge (CE) peer router. Use the **per-vrf** keyword to specify that the same label be used for all the routes advertised from a unique VRF.

Constrained Route Distribution for BGP/MPLS Internet Protocol VPNs

Constrained Route Distribution is a feature that service providers use in Multiprotocol Label Switching (MPLS) Layer 3 Virtual Private Networks (L3VPNs) to reduce the number of unnecessary routing updates that route reflectors (RR) send to provider edge (PE) routers. The reduction in routing updates saves resources. RRs, autonomous system boundary routers (ASBRs), and PEs will have fewer routes to carry. Route targets are used to constrain routing updates.

Some service providers have a very large number of routing updates being sent from RRs to PEs, using considerable resources. A PE does not need routing updates for VRFs that are not on the PE; therefore, the PE determines that many routing updates it receives are "unwanted." The PE filters out these unwanted updates.

Now consider a scenario where there are two RRs with another set of PEs. Not only are there unwanted routing updates from RR to PE, there are also unwanted routing updates between the RRs. As a result, a large number of unwanted routes might be advertised among RRs and PEs. The Constrained Route Distribution feature addresses this problem by filtering unwanted routing updates. When the Constrained Route Distribution is in place, the RR filters the updates.

Constrained Route Distribution Benefits

In MPLS L3VPNs, PE routers use BGP and Route Target (RT) extended communities to control the distribution of VPN routes, to and from VRFs, to separate the VPNs. It is common for PEs and Autonomous System Boundary Routers (ASBRs) to receive, and then filter out, unwanted VPN routes.

However, receiving and filtering unwanted VPN routes is a waste of resources. The sender generates and transmits a VPN routing update and the receiver filters out the unwanted routes. It would save resources to prevent, in the first place, the generation of such VPN route updates.

Address Family Route Target Filter (ARTF) is a mechanism that prevents the propagation of VPN Network Layer Reachability Information (NLRI) from the RR to a PE that is not interested in the VPN. This mechanism provides considerable savings in CPU cycles and transient memory usage. RT constraint limits the number of VPN routes and describes VPN membership.

BGP RT-constrain SAFI—rt-filter

The constrained route distribution feature introduces "rt-filter" subsequent address family identifier (SAFI), the BGP RT-constrain SAFI. Use the **address-family ipv4 rt-filter** command to enter the rt-filter SAFI. This SAFI carries route target (RT) filter information relevant to the BGP neighbor advertising it.

The Multiprotocol capability for ipv4 rt-filter address-family is advertised when the AFI is enabled under the neighbor. The rt-filter SAFI needs to be enabled globally, before it can be enabled under the neighbor. The rt-filter address family is allowed on both the iBGP and eBGP neighbors under default VRF.



Note If there are peers that are not RT-constrain capable, the RT-constrain address family must be enabled under all PE neighbors on RR. If all peers are RT-constrain capable, then the default RT-constrain route is not sent to the peers.

This example explains how to configure address-family ipv4 rt-filter:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#router bgp 100
RP/0/RP0/CPU0:router(config-bgp)#address-family ipv4 rt-filter
RP/0/RP0/CPU0:router(config-bgp-af)#neighbor 1.1.1.1
RP/0/RP0/CPU0:router(config-bgp-nbr)#address-family ipv4 rt-filter
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#
```

To get default rt-filter prefix information, use the **show bgp ipv4 rt-filter 0:2:0:0/0** command:

```
RP/0/RP0/CPU0:router#show bgp ipv4 rt-filter 0:2:0:0/0
BGP routing table entry for 0:2:0:0/0
Versions:
  Process          bRIB/RIB   SendTblVer
  Speaker          8489       8489
Last Modified: Jul  2 10:30:42.452 for 3d03h
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    0.0.0.0 from 0.0.0.0 (192.192.5.7)
    Received Label 1
    Origin IGP, localpref 100, valid, redistributed, best, group-best
    Received Path ID 0, Local Path ID 1, version 8489
```

Selective VRF Download

Selective VRF Download (SVD) feature enables the downloading of only those prefixes and labels to a line card that are actively required to forward traffic through the line card.

To meet the demand for a consolidated edge MSE platform, the number of VRFs, VRF interfaces, and the prefix capacity increase. Convergence timings differ in different line card engines. One of the major factors that determine convergence timing is the time taken to process and program a prefix and its associated data structures. A lesser number of prefixes and labels ensure better convergence timing. By enabling selective download of VRF routes to both Engine-3 (E3) and Engine-5 (E5) line cards, SVD reduces scalability and convergence problems in Layer 3 VPNs (L3VPNs)..

By default, SVD is enabled on the line cards. Use the **selective-vrf-download disable** command to disable SVD. Use the **show svd role** and **show svd state** commands to display role and state information of SVD on line cards.

Line Card Roles and Filters in Selective VRF Download

In a selective VRF download (SVD) context, line cards have these roles:

- Core LC: a line card that has only core facing interfaces (interfaces that connect to other P/PEs)
- Customer LC: a line card that has one or more customer facing interfaces (interfaces that connect to CEs in different VRFs)

The line cards handle these prefixes:

- Local Prefix: a prefix that is received from a CE connected to the router in a configured VRF context
- Remote Prefix: a prefix received from another PE and is imported to a configured VRF

These filters are applicable to each line card type:

- A core LC needs all the local prefixes and VRF labels so that the label or IP forwarding, or both is set up correctly.
- A customer LC needs both local and remote prefixes for all the VRFs to which it is connected, and for other VRFs which some connected VRFs have dependency. This is based on the import/export RT configuration; VRF 'A' may have imported routes from VRF 'B', so the imported route in VRF 'A' points to a next-hop that is in VRF 'B'. For route resolution, VRF 'B' routes need to be downloaded to each line card that has a VRF 'A' interface.
- If a line card hosts both core facing and customer facing interfaces, then it does not need to do any filtering. All tables and all routes are present on such line cards. These line cards have a role called "standard". All RPs and DRPs have the standard role.
- To correctly resolve L3VPN routes, the IPv4 default table needs to be present on all nodes. However, if the line card does not have any IPv6 interface, it can filter out all IPv6 tables and routes. In such a case, the line card can be deemed "not interested" in the IPv6 AFI. Then it behaves as if IPv6 is not supported by it.

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.

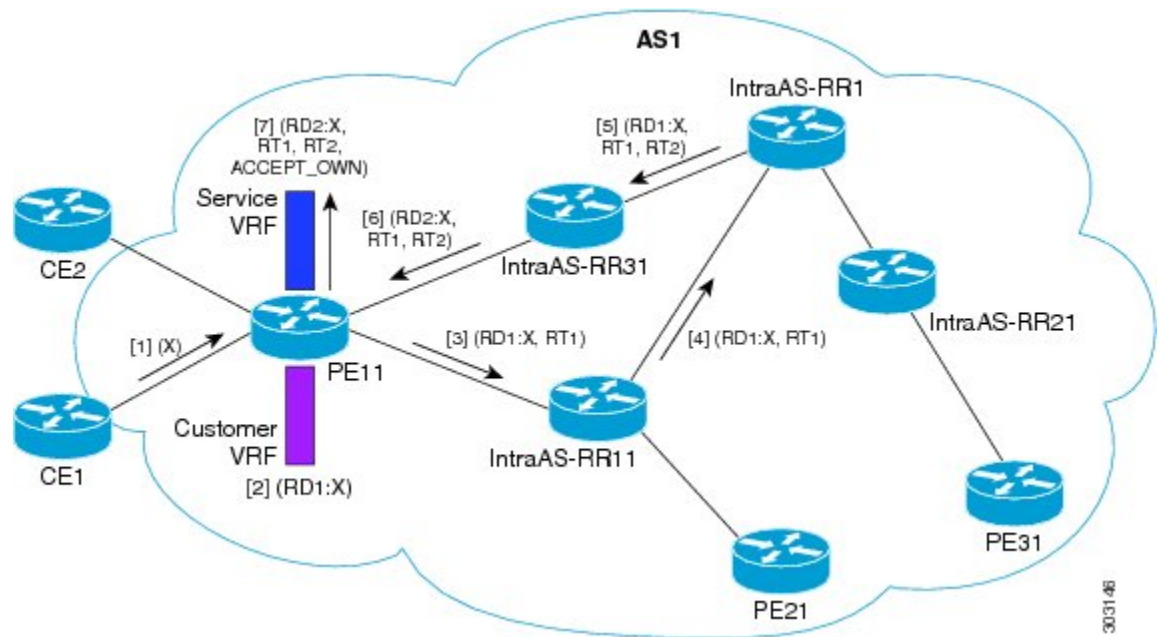
One of the applications of BGP Accept Own is auto-configuration of extranets within MPLS VPN networks. In an extranet configuration, routes present in one VRF is imported into another VRF on the same PE. Normally, the extranet mechanism requires that either the import-rt or the import policy of the extranet 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), performs 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.

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.

BFD Multihop Support for BGP

Bi-directional Forwarding Detection Multihop (BFD-MH) support is enabled for BGP. BFD Multihop establishes a BFD session between two addresses that may span multiple network hops. Cisco IOS XR Software BFD Multihop is based on RFC 5883. For more information on BFD Multihop, refer *Interface and Hardware Component Configuration Guide for Cisco CRS Routers* and *Interface and Hardware Component Command Reference for Cisco CRS Routers*.

BGP Multi-Instance and Multi-AS

Multiple BGP instances are supported on the router corresponding to a Autonomous System (AS). Each BGP instance is a separate process running on the same or on a different RP/DRP node. The BGP instances do not share any prefix table between them. No need for a common adj-rib-in (bRIB) as is the case with distributed BGP. The BGP instances do not communicate with each other and do not set up peering with each other. Each individual instance can set up peering with another router independently.

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.

- Cisco IOS XR CRS Multi-chassis systems can be used optimally by placing the different BGP instances on different RP/DRPs.
- 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

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.

Configuring 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

SUMMARY STEPS

1. `configure`
2. `router bgp as-number`

3. **rpki server** *{host-name | ip-address}*
4. Use one of these commands:
 - **transport ssh port** *port_number*
 - **transport tcp port** *port_number*
5. (Optional) **username** *user_name*
6. (Optional) **password** *password*
7. **preference** *preference_value*
8. **purge-time** *time*
9. Use one of these commands.
 - **refresh-time** *time*
 - **refresh-time off**
10. Use one these commands.
 - **response-time** *time*
 - **response-time off**
11. **commit**
12. (Optional) **shutdown**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 server <i>{host-name ip-address}</i> Example: RP/0/RP0/CPU0: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: <ul style="list-style-type: none"> • transport ssh port <i>port_number</i> • transport tcp port <i>port_number</i> Example: RP/0/RP0/CPU0:router(config-bgp-rpki-server)#transport ssh port 22 Or RP/0/RP0/CPU0:router(config-bgp-rpki-server)#transport tcp port 2	Specifies a transport method for the RPKI cache. <ul style="list-style-type: none"> • ssh—Select ssh to connect to the RPKI cache using SSH. • tcp—Select tcp to connect to the RPKI cache using TCP (unencrypted). • port <i>port_number</i>—Specify a port number for the specified RPKI cache transport. For tcp, the range of supported port number is 1 to 65535. For ssh, use port number 22. <p>Note Do not specify a custom port number for RPKI cache transport over SSH. You must use port 22 for RPKI over SSH.</p>

	Command or Action	Purpose
		Note You can set the transport to either TCP or SSH. Change of transport causes the cache session to flap.
Step 5	(Optional) username <i>user_name</i> Example: RP/0/RP0/CPU0:router (config-bgp-rpki-server) #username ssh_rpki_uname	Specifies a (SSH) username for the RPKI cache-server.
Step 6	(Optional) password <i>password</i> Example: RP/0/RP0/CPU0: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 <i>preference_value</i> Example: RP/0/RP0/CPU0: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 <i>time</i> Example: RP/0/RP0/CPU0: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. <ul style="list-style-type: none"> • refresh-time <i>time</i> • refresh-time off Example: RP/0/RP0/CPU0:router (config-bgp-rpki-server) #refresh-time 20 Or RP/0/RP0/CPU0: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. <ul style="list-style-type: none"> • response-time <i>time</i> • response-time off Example: RP/0/RP0/CPU0:router (config-bgp-rpki-server) #response-time 30 Or RP/0/RP0/CPU0: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	commit	

	Command or Action	Purpose
Step 12	(Optional) shutdown Example: RP/0/RP0/CPU0:router (config-bgp-rpki-server) #shutdown	Configures shut down of the RPKI cache.

Configuring RPKI Prefix Validation

Perform this task to control the behavior of RPKI prefix validation processing.

•

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. Use one of these commands.
 - **bgp origin-as validation disable**
 - **bgp origin-as validation time** {off | *prefix_validation_time*
4. **bgp origin-as validation signal** *ibgp*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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	Use one of these commands. <ul style="list-style-type: none"> • bgp origin-as validation disable • bgp origin-as validation time {off <i>prefix_validation_time</i> Example: RP/0/RP0/CPU0:router (config-bgp) #bgp origin-as validation disable Or RP/0/RP0/CPU0:router (config-bgp) #bgp origin-as validation time 50 Or	Sets the BGP origin-AS validation parameters. <ul style="list-style-type: none"> • disable—Use disable option to disable RPKI origin-AS validation. • time—Use time option to either set prefix validation time (in seconds) or to set off the automatic prefix validation after an RPKI update. Range for prefix validation time is 5 to 60 seconds. Configuring the disable option disables prefix validation for all eBGP paths and all eBGP paths are marked as "valid" by default.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-bgp)#bgp origin-as validation time off	Note The bgp origin-as validation options can also be configured in neighbor and neighbor address family submodes. The neighbor must be an eBGP neighbor. If configured at the neighbor or neighbor address family level, prefix validation disable or time options will be valid only for that specific neighbor or neighbor address family.
Step 4	bgp origin-as validation signal ibgp Example: RP/0/RP0/CPU0:router(config-bgp)#bgp origin-as validity signal ibgp	Enables the iBGP signaling of validity state through an extended-community. This can also be configured in global address family submode.
Step 5	commit	

Configure BGP Prefix Validation

Starting from Release 6.5.1, origin-as validation is disabled by default, you must enable it per address family. From Release 6.5.1, use the following task to configure RPKI Prefix Validation.

```
Router(config)# router bgp 100
/* The bgp origin-as validation time and bgp origin-as validity signal ibgp commands are optional. */
Router(config-bgp)# bgp origin-as validation time 50
Router(config-bgp)# bgp origin-as validation time off
Router(config-bgp)# bgp origin-as validation signal ibgp
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# bgp origin-as validation enable
```

Use the following commands to verify the origin-as validation configuration:

```
Router# show bgp origin-as validity

Thu Mar 14 04:18:09.656 PDT
BGP router identifier 10.1.1.1, local AS number 1
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000 RD version: 514
BGP main routing table version 514
BGP NSR Initial initsync version 2 (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
Origin-AS validation codes: V valid, I invalid, N not-found, D disabled
   Network                Next Hop                Metric LocPrf Weight Path
*> 209.165.200.223/27      0.0.0.0                  0           32768 ?
*> 209.165.200.225/27      0.0.0.0                  0           32768 ?
*> 19.1.2.0/24             0.0.0.0                  0           32768 ?
*> 19.1.3.0/24            0.0.0.0                  0           32768 ?
```

```

*> 10.1.2.0/24          0.0.0.0          0          32768 ?
*> 10.1.3.0/24          0.0.0.0          0          32768 ?
*> 10.1.4.0/24          0.0.0.0          0          32768 ?
*> 198.51.100.1/24     0.0.0.0          0          32768 ?
*> 203.0.113.235/24    0.0.0.0          0          32768 ?
V*> 209.165.201.0/27    10.1.2.1         0          4002 i
N*> 198.51.100.2/24    10.1.2.1         0          4002 i
I*> 198.51.100.1/24    10.1.2.1         0          4002 i
*> 192.0.2.1.0/24     0.0.0.0          0          32768 ?

Router# show bgp process
Mon Jul  9 16:47:39.428 PDT

BGP Process Information:
...
Use origin-AS validity in bestpath decisions
Allow (origin-AS) INVALID paths
Signal origin-AS validity state to neighbors

Address family: IPv4 Unicast
...
Origin-AS validation is enabled for this address-family
Use origin-AS validity in bestpath decisions for this address-family
Allow (origin-AS) INVALID paths for this address-family
Signal origin-AS validity state to neighbors with this address-family

```

Configuring RPKI Bestpath Computation

Perform this task to configure RPKI bestpath computation options.

SUMMARY STEPS

1. **configure**
2. **router bgp *as-number***
3. **bgp bestpath origin-as use validity**
4. **bgp bestpath origin-as allow invalid**
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)#router bgp 100	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

	Command or Action	Purpose
Step 3	bgp bestpath origin-as use validity Example: <pre>RP/0/RP0/CPU0:router(config-bgp)#bgp bestpath origin-as use validity</pre>	Enables the validity states of BGP paths to affect the path's preference in the BGP best path process. This configuration can also be done in router BGP address family submode.
Step 4	bgp bestpath origin-as allow invalid Example: <pre>RP/0/RP0/CPU0:router(config-bgp)#bgp bestpath origin-as allow invalid</pre>	<p>Allows all "invalid" paths to be considered for BGP bestpath computation.</p> <p>Note This configuration can also be done at global address family, neighbor, and neighbor address family submodes. Configuring <code>bgp bestpath origin-as allow invalid</code> in router BGP and address family submodes allow all "invalid" paths to be considered for BGP bestpath computation. By default, all such paths are not bestpath candidates. Configuring <code>bgp bestpath origin-as allow invalid</code> in neighbor and neighbor address family submodes allow all "invalid" paths from that specific neighbor or neighbor address family to be considered as bestpath candidates. The neighbor must be an eBGP neighbor.</p> <p>This configuration takes effect only when the bgp bestpath origin-as use validity configuration is enabled.</p>
Step 5	commit	

BGP 3107 PIC Updates for Global Prefixes

The BGP 3107 PIC Updates for Global Prefixes feature supports Prefix Independent Convergence (PIC) updates for global IPv4 and IPv6 prefixes in an MPLS VPN provider network. This feature is based on RFC 3107 that describes using BGP to distribute MPLS labels for global IPv4 or IPv6 prefixes. This enables IGP to scale better and also provides PIC updates for fast convergence.

BGP 3107 PIC is supported on CRS-1 and CRS-3 line cards.

RFC 3107 enables routes and labels to be carried in BGP. When BGP is used to distribute a particular route, it can also be used to distribute an MPLS label that is mapped to that route. The label mapping information for a particular route is piggybacked in the same BGP Update message that is used to distribute the route itself. RFC 3107 allows filtering of Next-Hop Loops from OSPF and reduces labels advertised by LDP. This implementation significantly reduces OSPF and LDP database.

The 3107 PIC implementation supports the following address-families with additional-path configuration.

- address-family ipv4 unicast
- address-family ipv6 unicast
- address-family vpv4 unicast
- address-family vpv6 unicast



Note The address-family l2vpn vpls-vpws does not support additional-path. Hence, the l2vpn service that uses address-family l2vpn vpls-vpws does not guarantee PIC convergence time.

The 3107 PIC implementation supports these Cisco IOS XR features:

- PIC Edge for 3107
- Traffic Engineering Fast-reroute (TE FRR)—Traffic convergence for core link failure is guaranteed within 50 milliseconds using verbatim tunnel.
- L2VPN Service (VPWS)
- L3VPN VPNv4 Service
- 6 PE Service
- 6 VPE Service
- VPLS Service

BGP 3107 PIC Updates for Global Prefixes implementation uses a shared recursive Load Info (RLDI) forwarding object in place of a Light-Weight recursive (LW-RLDI) object. The RLDI is shared between multiple leaves, while the LW-RLDI is instantiated per leaf. Sharing helps in handling PIC updates since it will be prefix independent.

BGP Prefix Independent Convergence for RIB and FIB

BGP PIC for RIB and FIB adds support for static recursive as PE-CE and faster backup activation by using fast re-route trigger.

The BGP PIC for RIB and FIB feature supports:

- FRR-like trigger for faster PE-CE link down detection, to further reduce the convergence time (Fast PIC-edge activation).
- PIC-edge for static recursive routes.
- BFD single-hop trigger for PIC-Edge without any explicit /32 static route configuration.
- Recursive PIC activation at third level and beyond, on failure trigger at the first (IGP) level.
- BGP path recursion constraints in FIB to ensure that FIB is in sync with BGP with respect to BGP next-hop resolution.

When BGP PIC Edge is configured, configuring the **neighbor shutdown** command does not trigger CEF to switch to the backup path. Instead, BGP starts to feed CEF again one by one from the top prefix of the routing table to the end thus causing a time delay.



Caution The time delay causes a black hole in the network. As a workaround, you must route the traffic to the backup path manually before configuring the **neighbor shutdown** command.

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 Attribute Filter Actions

The Attribute-filtering is configured by specifying a single or a range of attribute codes and an associated action. The allowed actions are:

- "Treat-as-withdraw"— The associated IPv4-unicast or MP_REACH NLRIs, if present, are withdrawn from the neighbor's Adj-RIB-In.
- "Discard Attribute"—The matching attributes alone are discarded and the rest of the Update message is processed normally.

When a received Update message contains one or more filtered attributes, the configured action is applied on the message. Optionally, the Update message is also stored to facilitate further debugging and a syslog message is generated on the console.

When an attribute matches the filter, further processing of the attribute is stopped and the corresponding action is taken.

Use the **attribute-filter group** command to enter Attribute-filter group command mode. Use the **attribute** command in attribute-filter group command mode to either discard an attribute or treat the update message as a "Withdraw" action.

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 0x00000840, action taken "TreatAsWithdraw".
Error details: "Error 0x00000800, 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 Link-State

BGP Link-State (LS) is an Address Family Identifier (AFI) and Sub-address Family Identifier (SAFI) defined to carry interior gateway protocol (IGP) link-state database through BGP. BGP LS delivers network topology information to topology servers and Application Layer Traffic Optimization (ALTO) servers. BGP LS allows policy-based control to aggregation, information-hiding, and abstraction. BGP LS supports IS-IS and OSPFv2.



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

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.

BGP VRF Dynamic Route Leaking

The Border Gateway Protocol (BGP) dynamic route leaking feature provides the ability to import routes between the default-vrf (Global VRF) and any other non-default VRF, to provide connectivity between a global and a VPN host. The import process installs the Internet route in a VRF table or a VRF route in the Internet table, providing connectivity.



Note

- Directly connected routes cannot be leaked using BGP VRF Dynamic Route Leaking from default VRF to non-default VRF

The dynamic route leaking is enabled by:

- Importing from default-VRF to non-default-VRF, using the **import from default-vrf route-policy route-policy-name [advertise-as-vpn]** command in VRF address-family configuration mode.

If the **advertise-as-vpn** option is configured, the paths imported from the default-VRF to the non-default-VRF are advertised to the PEs as well as to the CEs. If the **advertise-as-vpn** option is not configured, the paths imported from the default-VRF to the non-default-VRF are not advertised to the PE. However, the paths are still advertised to the CEs.

- Importing from non-default-VRF to default VRF, using the **export to default-vrf route-policy route-policy-name** command in VRF address-family configuration mode.

A route-policy is mandatory to filter the imported routes. This reduces the risk of unintended import of routes between the Internet table and the VRF tables and the corresponding security issues.

There is no hard limit on the number of prefixes that can be imported. The import creates a new prefix in the destination VRF, which increases the total number of prefixes and paths. However, each VRF importing global routes adds workload equivalent to a neighbor receiving the global table. This is true even if the user filters out all but a few prefixes. Hence, importing five to ten VRFs is ideal.

Resilient Per-CE Label Mode

The Resilient Per-CE Label is an extension of the Per-CE label mode to support Prefix Independent Convergence (PIC) and load balancing.

At present, the three label modes, Per-Prefix, Per-CE, and Per-VRF have these restrictions:

- Resilient per-CE 6PE label is not supported on CRS-1 and CRS-3 routers, but supported only on CRS-X routers
- No support for PIC
- No support for load balancing across CEs
- Temporary forwarding loop during local traffic diversion to support PIC
- No support for EIBGP multipath load balancing
- Forwarding performance impact
- Per-prefix label mode causes scale issues on another vendor router in a network

In the Resilient Per-CE label scheme, BGP installs a unique rewrite label in LSD for every unique set of CE paths or next hops. There may be one or more prefixes in BGP table that points to this label. BGP also installs the CE paths (primary) and optionally a backup PE path into RIB. FIB learns about the label rewrite information from LSD and the IP paths from RIB.

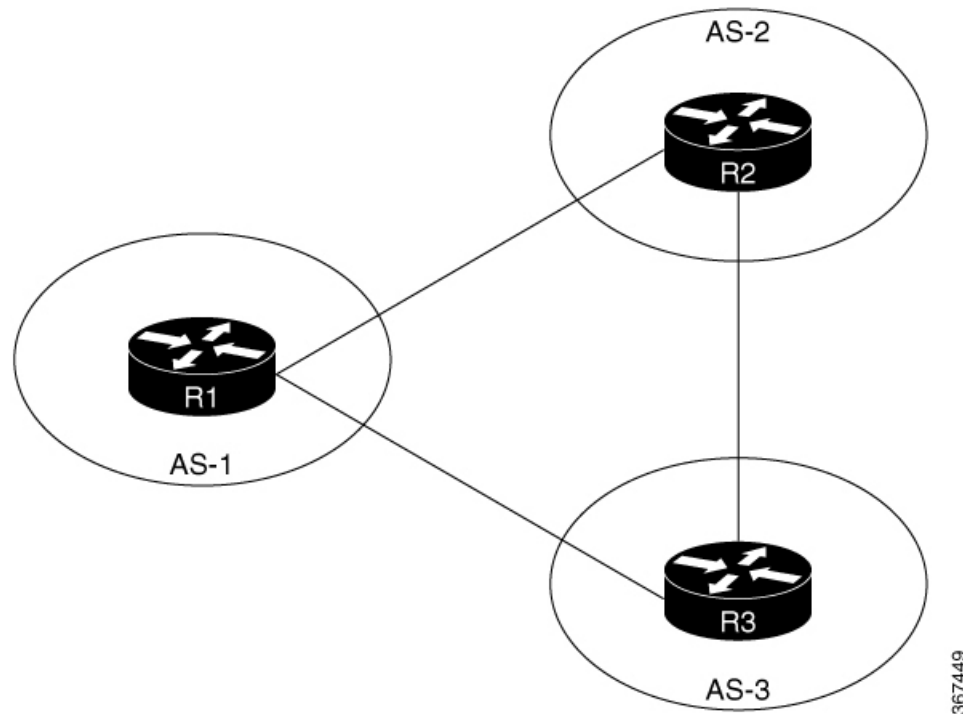
In steady state, labeled traffic destined to the resilient per-CE label is load balanced across all the CE next hops. When all the CE paths fail, any traffic destined to that label will result in an IP lookup and will be forwarded towards the backup PE path, if available. This action is performed on the label independently of the number of prefixes that may point to the label, resulting in the PIC behavior during primary paths failure.

BGP Multipath Enhancements

- Overwriting of next-hop calculation for multipath prefixes is not allowed. The **next-hop-unchanged multipath** command disables overwriting of next-hop calculation for multipath prefixes.
- 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 13: 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.

MVPN with BGP SAFI-2 and SAFI-129

BGP supports Subsequent Address Family Identifier (SAFI)-2 and SAFI-129 for multicast VPNs (MVPNs).

SAFI-129 provides the capability to support multicast routing in the core IPv4 network. SAFI-129 supports BGP-based MVPNs. The addition of SAFI-129 allows multicast to select an upstream multicast hop that may be independent of the unicast topology. Multicast routes learned from the customer edge (CE) router or multicast VPN routes learned from remote provider edge (PE) routers are installed into the multicast Routing Information Base (MuRIB). This MuRIB will be populated with routes that are specific to multicast, and are not used by unicast forwarding. The PE-CE BGP prefixes are advertised using SAFI-2, the PE-PE routes are advertised using SAFI-129.

How to Implement BGP

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 While establishing eBGP neighborhood 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.

SUMMARY STEPS

1. **configure**
2. **route-policy** *route-policy-name*
3. **end-policy**
4. **commit**
5. **configure**
6. **router bgp** *as-number*
7. **bgp router-id** *ip-address*

8. **address-family** { ipv4 | ipv6 } unicast
9. **exit**
10. **neighbor** *ip-address*
11. **remote-as** *as-number*
12. **address-family** { ipv4 | ipv6 } unicast
13. **route-policy** *route-policy-name* { in | out }
14. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	route-policy <i>route-policy-name</i> Example: <pre>RP/0/RP0/CPU0:router(config)# route-policy drop-as-1234 RP/0/RP0/CPU0:router(config-rpl)# if as-path passes-through '1234' then RP/0/RP0/CPU0:router(config-rpl)# apply check-communities RP/0/RP0/CPU0:router(config-rpl)# else RP/0/RP0/CPU0:router(config-rpl)# pass RP/0/RP0/CPU0:router(config-rpl)# endif</pre>	(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.
Step 3	end-policy Example: <pre>RP/0/RP0/CPU0:router(config-rpl)# end-policy</pre>	(Optional) Ends the definition of a route policy and exits route policy configuration mode.
Step 4	commit	
Step 5	configure	
Step 6	router bgp <i>as-number</i> Example: <pre>RP/0/RP0/CPU0: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 7	bgp router-id <i>ip-address</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 192.168.70.24</pre>	Configures the local router with a specified router ID.
Step 8	address-family { ipv4 ipv6 } unicast Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast</pre>	<p>Specifies either the IPv4 or IPv6 address family and enters address family configuration submenu.</p> <p>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</p>

	Command or Action	Purpose
Step 9	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.
Step 10	neighbor ip-address Example: RP/0/RP0/CPU0: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 11	remote-as as-number Example: RP/0/RP0/CPU0:router(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: 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 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: RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy drop-as-1234 in	(Optional) Applies the specified policy to inbound IPv4 unicast routes.
Step 14	commit	

Configuring 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> [instance <i>instance name</i>] Example: RP/0/RSP0/CPU0:router(config)# router bgp 100 instance inst1	Enters BGP configuration mode for the user specified BGP instance.
Step 3	bgp router-id <i>ip-address</i> Example: RP/0/RSP0/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	commit	

Configuring a 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.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp confederation identifier** *as-number*
4. **bgp confederation peers** *as-number*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 <i>as-number</i> Example:	Specifies a BGP confederation identifier.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-bgp)# bgp confederation identifier 5	
Step 4	<p>bgp confederation peers <i>as-number</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 1091 RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 1092 RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 1093 RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 1094 RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 1095 RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 1096</pre>	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	commit	

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:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#lpts pifib hardware police location 0/2/CPU0
RP/0/RP0/CPU0:router(config-pifib-policer-per-node)#flow bgp configured rate 4000
RP/0/RP0/CPU0:router(config-pifib-policer-per-node)#flow bgp known rate 4000
RP/0/RP0/CPU0:router(config-pifib-policer-per-node)#flow bgp default rate 4000
RP/0/RP0/CPU0:router(config-pifib-policer-per-node)#commit
```

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.

Adjusting BGP Timers

Perform this task to set the timers for BGP neighbors.

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.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **timers bgp** *keepalive hold-time*
4. **neighbor** *ip-address*
5. **timers** *keepalive hold-time*
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 <i>keepalive hold-time</i> Example: RP/0/RP0/CPU0:router(config-bgp)# timers bgp 30 90	Sets a default keepalive time and a default hold time for all neighbors.
Step 4	neighbor <i>ip-address</i> Example: RP/0/RP0/CPU0: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 <i>keepalive hold-time</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# timers 60 220	(Optional) Sets the keepalive timer and the hold-time timer for the BGP neighbor.
Step 6	commit	

Changing the BGP Default Local Preference Value

Perform this task to set the default local preference value for BGP paths.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*

3. `bgp default local-preference value`
4. `commit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 <i>value</i> Example: RP/0/RP0/CPU0: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	<code>commit</code>	

Configuring the 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).

SUMMARY STEPS

1. `configure`
2. `router bgp as-number`
3. `default-metric value`
4. `commit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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	default-metric <i>value</i> Example: RP/0/RP0/CPU0:router(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).

	Command or Action	Purpose
Step 4	commit	

Configuring BGP Weights

Perform this task to assign a weight to routes received from a neighbor. 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.

Before you begin



Note The **clear bgp** command must be used for the newly configured weight to take effect.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **address-family** { *ipv4* | *ipv6* } **unicast**
6. **weight** *weight-value*
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 <i>ip-address</i> Example: RP/0/RP0/CPU0: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 <i>as-number</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002	Creates a neighbor and assigns a remote autonomous system number to it.

	Command or Action	Purpose
Step 5	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 submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 6	weight <i>weight-value</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr-af)# weight 41150	Assigns a weight to all routes learned through the neighbor.
Step 7	commit	

Tuning the BGP Best-Path Calculation

Perform this task to change the default BGP best-path calculation behavior.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp bestpath med missing-as-worst**
4. **bgp bestpath med always**
5. **bgp bestpath med confed**
6. **bgp bestpath as-path ignore**
7. **bgp bestpath compare-routerid**
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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:	Configures the BGP speaker in the specified autonomous system to compare MEDs among all the paths for the prefix,

	Command or Action	Purpose
	<code>RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath med always</code>	regardless of the autonomous system from which the paths are received.
Step 5	bgp bestpath med confed Example: <code>RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath med confed</code>	Enables BGP software to compare MED values for paths learned from confederation peers.
Step 6	bgp bestpath as-path ignore Example: <code>RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath as-path ignore</code>	Configures the BGP software to ignore the autonomous system length when performing best-path selection.
Step 7	bgp bestpath compare-routerid Example: <code>RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath compare-routerid</code>	Configure the BGP speaker in the autonomous system to compare the router IDs of similar paths.
Step 8	commit	

Indicating 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.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { *ipv4* | *ipv6* } **unicast**
4. **network** { *ip-address / prefix-length* | *ip-address mask* } **backdoor**
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: <code>RP/0/RP0/CPU0:router(config)# router bgp 120</code>	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	address-family { ipv4 ipv6 } unicast Example: RP/0/RP0/CPU0: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 { <i>ip-address / prefix-length</i> <i>ip-address mask</i> } backdoor Example: RP/0/RP0/CPU0:router(config-bgp-af)# network 172.20.0.0/16	Configures the local router to originate and advertise the specified network.
Step 5	commit	

Configuring Aggregate Addresses

Perform this task to create aggregate entries in a BGP routing table.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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 <i>address/mask-length</i> [as-set] [as-confed-set] [summary-only] [route-policy <i>route-policy-name</i>]	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.

	Command or Action	Purpose
	<p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-af)# aggregate-address 10.0.0.0/8 as-set</pre>	<ul style="list-style-type: none"> • 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 <i>route-policy-name</i> keyword and argument specify the route policy used to set the attributes of the aggregate route.
Step 5	commit	

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.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp redistribute-internal**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	<p>router bgp <i>as-number</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0: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.

	Command or Action	Purpose
Step 3	bgp redistribute-internal Example: RP/0/RP0/CPU0:router(config-bgp)# bgp redistribute-internal	Allows the redistribution of iBGP routes into an IGP, such as IS-IS or OSPF.
Step 4	commit	

Configuring Discard Extra Paths

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: RP/0/RP0/CPU0:router# configure	Enters Global Configuration mode.
Step 2	router bgp <i>as-number</i> 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 <i>ip-address</i> 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 submode.
Step 5	maximum-prefix <i>maximum</i> 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.

	Command or Action	Purpose
Step 6	commit	

Configuring Per Neighbor TCP MSS

Perform this task to configure TCP MSS under neighbor group, which is inherited by a neighbor.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family ipv4 unicast**
4. **exit**
5. **neighbor-group** *name*
6. **tcp mss** *segment-size*
7. **address-family ipv4 unicast**
8. **exit**
9. **exit**
10. **neighbor** *ip-address*
11. **remote-as** *as-number*
12. **use neighbor-group** *group-name*
13. **address-family ipv4 unicast**
14. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: RP/0/RP0/CPU0:router# configure	Enters Global Configuration mode.
Step 2	router bgp <i>as-number</i> 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	address-family ipv4 unicast Example: RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast	Specifies the IPv4 address family unicast and enters address family configuration mode.
Step 4	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits router address family configuration mode, and returns to BGP configuration mode.

	Command or Action	Purpose
Step 5	neighbor-group <i>name</i> Example: RP/0/RP0/CPU0:router(config-bgp)# neighbor-group n1	Enters neighbor group configuration mode.
Step 6	tcp mss <i>segment-size</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast	Specifies the IPv4 address family unicast and enters address family configuration mode.
Step 8	exit Example: RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit	Exits router address family configuration mode.
Step 9	exit Example: RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit	Exits the neighbor group configuration mode.
Step 10	neighbor <i>ip-address</i> Example: RP/0/RP0/CPU0: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 <i>as-number</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1	Creates a neighbor and assigns a remote autonomous system (AS) number to it. <ul style="list-style-type: none"> • 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 <i>group-name</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group n1	Specifies that the BGP neighbor inherit configuration from the specified neighbor group.

	Command or Action	Purpose
Step 13	address-family ipv4 unicast Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast RP/0/RP0/CPU0:router(config-bgp-nbr-af)#	Specifies the IPv4 address family unicast and enters address family configuration mode.
Step 14	commit	

Disabling Per Neighbor TCP MSS

Perform this task to disable TCP MSS for a particular neighbor under neighbor group.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family ipv4 unicast**
4. **exit**
5. **neighbor-group** *name*
6. **tcp mss** *segment-size*
7. **address-family ipv4 unicast**
8. **exit**
9. **exit**
10. **neighbor** *ip-address*
11. **remote-as** *as-number*
12. **use neighbor-group** *group-name*
13. **tcp mss inheritance-disable**
14. **address-family ipv4 unicast**
15. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: RP/0/RP0/CPU0:router# configure	Enters Global Configuration mode.
Step 2	router bgp <i>as-number</i> 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	address-family ipv4 unicast Example:	Specifies the IPv4 address family unicast and enters address family configuration mode.

	Command or Action	Purpose
	<pre>RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast</pre>	
Step 4	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-af)# exit</pre>	Exits router address family configuration mode, and returns to BGP configuration mode.
Step 5	neighbor-group name Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# neighbor-group n1</pre>	Enters neighbor group configuration mode.
Step 6	tcp mss segment-size Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# tcp mss 500</pre>	Configures TCP maximum segment size. The range is from 68 to 10000.
Step 7	address-family ipv4 unicast Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast</pre>	Specifies the IPv4 address family unicast and enters address family configuration mode.
Step 8	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit</pre>	Exits router address family configuration mode.
Step 9	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit</pre>	Exits the neighbor group configuration mode.
Step 10	neighbor ip-address Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.2</pre>	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: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1</pre>	Creates a neighbor and assigns a remote autonomous system (AS) number to it. <ul style="list-style-type: none"> • 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.

	Command or Action	Purpose
Step 12	use neighbor-group <i>group-name</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group n1</pre>	Specifies that the BGP neighbor inherit configuration from the specified neighbor group.
Step 13	tcp mss inheritance-disable Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# tcp mss inheritance-disable</pre>	Disables TCP MSS for the neighbor.
Step 14	address-family ipv4 unicast Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast RP/0/RP0/CPU0:router(config-bgp-nbr-af)#</pre>	Specifies the IPv4 address family unicast and enters address family configuration mode.
Step 15	commit	

Redistributing Prefixes into Multiprotocol BGP

Perform this task to redistribute prefixes from another protocol into multiprotocol BGP.

Redistribution is the process of injecting prefixes from one routing protocol into another routing protocol. This task shows how to inject prefixes from another routing protocol into multiprotocol BGP. Specifically, prefixes that are redistributed into multiprotocol BGP using the **redistribute** command are injected into the unicast database, the multicast database, or both.



Note BGP doesn't support redistribution of ISIS routes in VRF.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. 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 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*]

5. commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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	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 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>] • redistribute static [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] Example: RP/0/RP0/CPU0:router(config-bgp-af)# redistribute ospf 110	Causes routes from the specified instance to be redistributed into BGP.
Step 5	commit	

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. **commit**
6. **show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6** { **unicast** | **multicast** | **all** | **tunnel** } | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpn4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**] | **vpn6 unicast** [**rd** *rd-address*]] **flap-statistics**
7. **show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6** { **unicast** | **multicast** | **all** | **tunnel** } | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpn4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**] | **vpn6 unicast** [**rd** *rd-address*]] **flap-statistics** **regex** *regular-expression*
8. **show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6** { **unicast** | **multicast** | **all** | **tunnel** } | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpn4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**] | **vpn6 unicast** [**rd** *rd-address*]] **route-policy** *route-policy-name*
9. **show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6** { **unicast** | **multicast** | **all** | **tunnel** } | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpn4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**] | **vpn6 unicast** [**rd** *rd-address*]] { *mask* | */prefix-length* }
10. **show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6** { **unicast** | **multicast** | **all** | **tunnel** } | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpn4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**] | **vpn6 unicast** [**rd** *rd-address*]] **flap-statistics** { *ip-address* [{ *mask* | */prefix-length* }] [**longer-prefixes**]
11. **clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6** { **unicast** | **multicast** | **all** | **tunnel** } | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpn4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**] | **vpn6 unicast** [**rd** *rd-address*]] **flap-statistics**
12. **clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6** { **unicast** | **multicast** | **all** | **tunnel** } | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpn4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**] | **vpn6 unicast** [**rd** *rd-address*]] **flap-statistics** **regex** *regular-expression*
13. **clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6** { **unicast** | **multicast** | **all** | **tunnel** } | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpn4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**] | **vpn6 unicast** [**rd** *rd-address*]] **route-policy** *route-policy-name*
14. **clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6** { **unicast** | **multicast** | **all** | **tunnel** } | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpn4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**] | **vpn6 unicast** [**rd** *rd-address*]] **flap-statistics** *network* / *mask-length*
15. **clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6** { **unicast** | **multicast** | **all** | **tunnel** } | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpn4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**] | **vpn6 unicast** [**rd** *rd-address*]] **flap-statistics** *ip-address* / *mask-length*

16. **show bgp** [ipv4 { unicast | multicast | labeled-unicast | all } | ipv6 { unicast | multicast | all | tunnel } | all { unicast | multicast | all | labeled-unicast } | vpnv4 unicast [rd rd-address] | vrf { vrf-name | all } [ipv4 { unicast | labeled-unicast } | ipv6 unicast] | vpnv6 unicast [rd rd-address]] dampened-paths
17. **clear bgp** [ipv4 { unicast | multicast | labeled-unicast | all } | ipv6 { unicast | multicast | all | tunnel } | all { unicast | multicast | all | labeled-unicast } | vpnv4 unicast [rd rd-address] | vrf { vrf-name | all } [ipv4 { unicast | labeled-unicast } | ipv6 unicast] | vpnv6 unicast [rd rd-address]] dampening ip-address / mask-length

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast	Specifies either the IPv4 or IPv6 address family 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	bgp dampening [<i>half-life</i> [<i>reuse suppress</i> <i>max-suppress-time</i>]] route-policy <i>route-policy-name</i>] Example: RP/0/RP0/CPU0:router(config-bgp-af)# bgp dampening 30 1500 10000 120	Configures BGP dampening for the specified address family.
Step 5	commit	
Step 6	show bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpnv4 unicast [rd rd-address] vrf { vrf-name all } [ipv4 { unicast labeled-unicast } ipv6 unicast] vpnv6 unicast [rd rd-address]] flap-statistics Example: RP/0/RP0/CPU0:router# show bgp flap statistics	Displays BGP flap statistics.
Step 7	show bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpnv4 unicast [rd rd-address] vrf { vrf-name all } [ipv4 { unicast labeled-unicast	Displays BGP flap statistics for all paths that match the regular expression.

	Command or Action	Purpose
	<pre> } ipv6 unicast] vpv6 unicast [rd rd-address]] flap-statistics regexp regular-expression Example: RP/0/RP0/CPU0:router# show bgp flap-statistics regexp _1\$ </pre>	
Step 8	<pre> show bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpv4 unicast [rd rd-address] vrf { vrf-name all } [ipv4 { unicast labeled-unicast } ipv6 unicast] vpv6 unicast [rd rd-address]] route-policy route-policy-name Example: RP/0/RP0/CPU0:router(config)# show bgp flap-statistics route-policy policy_A </pre>	Displays BGP flap statistics for the specified route policy.
Step 9	<pre> show bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpv4 unicast [rd rd-address] vrf { vrf-name all } [ipv4 { unicast labeled-unicast } ipv6 unicast] vpv6 unicast [rd rd-address]] { mask /prefix-length } Example: RP/0/RP0/CPU0:router# show bgp flap-statistics 172.20.1.1 </pre>	Displays BGP flap for the specified prefix.
Step 10	<pre> show bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpv4 unicast [rd rd-address] vrf { vrf-name all } [ipv4 { unicast labeled-unicast } ipv6 unicast] vpv6 unicast [rd rd-address]] flap-statistics { ip-address [{ mask /prefix-length } [longer-prefixes Example: RP/0/RP0/CPU0:router# show bgp flap-statistics 172.20.1.1 longer-prefixes </pre>	Displays BGP flap statistics for more specific entries for the specified IP address.
Step 11	<pre> clear bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpv4 unicast [rd rd-address] vrf { vrf-name all </pre>	Clears BGP flap statistics for all routes.

	Command or Action	Purpose
	<pre> } [ipv4 { unicast labeled-unicast } ipv6 unicast] vpnv6 unicast [rd rd-address]] flap-statistics Example: RP/0/RP0/CPU0:router# clear bgp all all flap-statistics </pre>	
Step 12	<pre> clear bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpnv4 unicast [rd rd-address] vrf { vrf-name all } [ipv4 { unicast labeled-unicast } ipv6 unicast] vpnv6 unicast [rd rd-address]] flap-statistics regex <i>regular-expression</i> Example: RP/0/RP0/CPU0:router# clear bgp ipv4 unicast flap-statistics regex _1\$ </pre>	Clears BGP flap statistics for all paths that match the specified regular expression.
Step 13	<pre> clear bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpnv4 unicast [rd rd-address] vrf { vrf-name all } [ipv4 { unicast labeled-unicast } ipv6 unicast] vpnv6 unicast [rd rd-address]] route-policy <i>route-policy-name</i> Example: RP/0/RP0/CPU0:router# clear bgp ipv4 unicast flap-statistics route-policy policy_A </pre>	Clears BGP flap statistics for the specified route policy.
Step 14	<pre> clear bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpnv4 unicast [rd rd-address] vrf { vrf-name all } [ipv4 { unicast labeled-unicast } ipv6 unicast] vpnv6 unicast [rd rd-address]] flap-statistics <i>network / mask-length</i> Example: RP/0/RP0/CPU0:router# clear bgp ipv4 unicast flap-statistics 192.168.40.0/24 </pre>	Clears BGP flap statistics for the specified network.
Step 15	<pre> clear bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpnv4 unicast [rd rd-address] vrf { vrf-name all } [ipv4 { unicast labeled-unicast } ipv6 unicast] </pre>	Clears BGP flap statistics for routes received from the specified neighbor.

	Command or Action	Purpose
	<p> vpnv6 unicast [rd <i>rd-address</i>]] flap-statistics <i>ip-address / mask-length</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# clear bgp ipv4 unicast flap-statistics 172.20.1.1</pre>	
Step 16	<p>show bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpnv4 unicast [rd <i>rd-address</i>] vrf { <i>vrf-name</i> all } [ipv4 { unicast labeled-unicast } ipv6 unicast] vpnv6 unicast [rd <i>rd-address</i>]] dampened-paths</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show bgp dampened-paths</pre>	Displays the dampened routes, including the time remaining before they are unsuppressed.
Step 17	<p>clear bgp [ipv4 { unicast multicast labeled-unicast all } ipv6 { unicast multicast all tunnel } all { unicast multicast all labeled-unicast } vpnv4 unicast [rd <i>rd-address</i>] vrf { <i>vrf-name</i> all } [ipv4 { unicast labeled-unicast } ipv6 unicast] vpnv6 unicast [rd <i>rd-address</i>]] dampening <i>ip-address / mask-length</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# clear bgp dampening</pre>	<p>Clears route dampening information and unsuppresses the suppressed routes.</p> <p>Caution Always use the clear bgp dampening command for an individual address-family. The all option for address-families with clear bgp dampening should never be used during normal functioning of the system. For example, use <code>clear bgp ipv4 unicast dampening prefix x.x.x./y</code></p>

Applying Policy When Updating the Routing Table

Perform this task to apply a routing policy to routes being installed into the routing table.

Before you begin

See the *Implementing Routing Policy on Cisco IOS XR Software* module of *Routing Configuration Guide for Cisco CRS Routers* (this publication) for a list of the supported attributes and operations that are valid for table policy filtering.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **table-policy** *policy-name*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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 <i>policy-name</i> Example: RP/0/RP0/CPU0:router(config-bgp-af)# table-policy tbl-plcy-A	Applies the specified policy to routes being installed into the routing table.
Step 5	commit	

Setting BGP Administrative Distance

Perform this task to specify the use of administrative distances that can be used to prefer one class of route over another.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **distance bgp** *external-distance internal-distance local-distance*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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:	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

	Command or Action	Purpose
	<pre>RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast</pre>	To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	<p>distance bgp <i>external-distance internal-distance local-distance</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-af)# distance bgp 20 20 200</pre>	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	commit	

Configuring a 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.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **exit**
5. **neighbor-group** *name*
6. **remote-as** *as-number*
7. **address-family** { **ipv4** | **ipv6** } **unicast**
8. **route-policy** *route-policy-name* { **in** | **out** }
9. **exit**

10. **exit**
11. **neighbor** *ip-address*
12. **use neighbor-group** *group-name*
13. **remote-as** *as-number*
14. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.
Step 5	neighbor-group <i>name</i> Example: RP/0/RP0/CPU0:router(config-bgp)# neighbor-group nbr-grp-A	Places the router in neighbor group configuration mode.
Step 6	remote-as <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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 <i>route-policy-name { in out }</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# route-policy drop-as-1234 in	(Optional) Applies the specified policy to inbound IPv4 unicast routes.

	Command or Action	Purpose
Step 9	exit Example: RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af) # exit	Exits the current configuration mode.
Step 10	exit Example: RP/0/RP0/CPU0:router(config-bgp-nbrgrp) # exit	Exits the current configuration mode.
Step 11	neighbor ip-address Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router(config-bgp-nbr) # remote-as 2002	Creates a neighbor and assigns a remote autonomous system number to it.
Step 14	commit	

Configuring a 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.

SUMMARY STEPS

1. **configure**
2. **router bgp as-number**
3. **bgp cluster-id cluster-id**

4. **neighbor** *ip-address*
5. **remote-as** *as-number*
6. **address-family** { *ipv4* | *ipv6* } **unicast**
7. **route-reflector-client**
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 <i>cluster-id</i> Example: RP/0/RP0/CPU0: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 <i>ip-address</i> Example: RP/0/RP0/CPU0: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 <i>as-number</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2003	Creates a neighbor and assigns a remote autonomous system number to it.
Step 6	address-family { <i>ipv4</i> <i>ipv6</i> } unicast Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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	commit	

Configuring BGP Route Filtering by Route Policy

Perform this task to configure BGP routing filtering by route policy.

Before you begin

See the *Implementing Routing Policy on Cisco IOS XR Software* module of *Cisco Cisco IOS XR Routing Configuration Guide* (this publication) for a list of the supported attributes and operations that are valid for inbound and outbound neighbor policy filtering.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **end-policy**
4. **router bgp** *as-number*
5. **neighbor** *ip-address*
6. **address-family** { **ipv4** | **ipv6** } **unicast**
7. **route-policy** *route-policy-name* { **in** | **out** }
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	route-policy <i>name</i> Example: <pre>RP/0/RP0/CPU0:router(config)# route-policy drop-as-1234 RP/0/RP0/CPU0:router(config-rpl)# if as-path passes-through '1234' then RP/0/RP0/CPU0:router(config-rpl)# apply check-communities RP/0/RP0/CPU0:router(config-rpl)# else RP/0/RP0/CPU0:router(config-rpl)# pass RP/0/RP0/CPU0:router(config-rpl)# endif</pre>	(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.
Step 3	end-policy Example: <pre>RP/0/RP0/CPU0:router(config-rpl)# end-policy</pre>	(Optional) Ends the definition of a route policy and exits route policy configuration mode.
Step 4	router bgp <i>as-number</i> Example: <pre>RP/0/RP0/CPU0: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.

	Command or Action	Purpose
Step 5	neighbor <i>ip-address</i> Example: RP/0/RP0/CPU0: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 { <i>ipv4</i> <i>ipv6</i> } unicast Example: RP/0/RP0/CPU0: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 <i>route-policy-name</i> { in out } Example: RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy drop-as-1234 in	Applies the specified policy to inbound routes.
Step 8	commit	

Configuring BGP Attribute Filtering

Perform the following tasks to 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

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 <i>attribute-filter group name</i> Example: RP/0/RP0/CPU0: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.

	Command or Action	Purpose
Step 4	<p>attribute <i>attribute code</i> { discard treat-as-withdraw }</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-attrfg)# attribute 24 discard</pre>	<p>Specifies a single or a range of attribute codes and an associated action. The allowed actions are:</p> <ul style="list-style-type: none"> • 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.

Configuring 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	<p>router bgp <i>as-number</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0: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	<p>address-family { ipv4 ipv6 } unicast</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast</pre>	<p>Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.</p> <p>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</p>
Step 4	<p>nexthop trigger-delay { critical <i>delay</i> non-critical <i>delay</i> }</p> <p>Example:</p>	Sets the critical next-hop trigger delay.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-bgp-af)# nexthop trigger-delay critical 15000	
Step 5	commit	

Disabling 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.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **address-family** { **ipv4** | **ipv6** } **unicast**
6. **next-hop-self**
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 <i>ip-address</i> Example: RP/0/RP0/CPU0: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 <i>as-number</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 206	Creates a neighbor and assigns a remote autonomous system number to it.

	Command or Action	Purpose
Step 5	address-family { ipv4 ipv6 } unicast Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast</pre>	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: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# next-hop-self</pre>	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	commit	

Configuring 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.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **address-family** { **ipv4** { **labeled-unicast** | **unicast** | **mdt** | **multicast** | **mvpn** | **rt-filter** | **tunnel** } | **ipv6** { **labeled-unicast** | **mvpn** | **unicast** } }
6. Use one of these commands:
 - **send-community-ebgp**
 - **send-extended-community-ebgp**
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 <i>ip-address</i> Example: RP/0/RP0/CPU0: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 <i>as-number</i> Example: RP/0/RP0/CPU0: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 multicast mvpn rt-filter tunnel } ipv6 { labeled-unicast mvpn unicast }} Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv6 unicast	<p>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.</p> <p>IPv6 address family mode supports these sub modes:</p> <ul style="list-style-type: none"> • labeled-unicast • mvpn • unicast <p>IPv4 address family mode supports these sub modes:</p> <ul style="list-style-type: none"> • labeled-unicast • mdt • multicast • mvpn • rt-filter • tunnel • unicast <p>Refer the address-family (BGP) command in <i>BGP Commands</i> module of <i>Routing Command Reference for Cisco CRS Routers</i> for more information on the Address Family Submode support.</p>

	Command or Action	Purpose
Step 6	Use one of these commands: <ul style="list-style-type: none"> • send-community-ebgp • send-extended-community-ebgp Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# send-community-ebgp</pre> or <pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# send-extended-community-ebgp</pre>	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	commit	

Configuring the BGP Cost Community

Perform this task to configure the 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.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **set extcommunity cost** { *cost-extcommunity-set-name* | *cost-inline-extcommunity-set* } [**additive**]
4. **end-policy**
5. **router bgp** *as-number*
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*]
 - **address-family** { *ipv4 unicast* | *ipv4 multicast* | *ipv4 tunnel* | *ipv4 mdt* | *ipv6 unicast* | *ipv6 multicast* | *vpn4 unicast* | *vpn6 unicast* } **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
 - **address-family** { *ipv4 unicast* | *ipv4 multicast* | *ipv4 tunnel* | *ipv4 mdt* | *ipv6 unicast* | *ipv6 multicast* | *vpn4 unicast* | *vpn6 unicast* } **redistribute eigrp** *process-id* [**match** { *external* | *internal* }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
 - **address-family** { *ipv4 unicast* | *ipv4 multicast* | *ipv4 tunnel* | *ipv4 mdt* | *ipv6 unicast* | *ipv6 multicast* | *vpn4 unicast* | *vpn6 unicast* } **redistribute isis** *process-id* [**level** { *1* | *1-inter-area* | *2* }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
 - **address-family** { *ipv4 unicast* | *ipv4 multicast* | *ipv4 tunnel* | *ipv4 mdt* | *ipv6 unicast* | *ipv6 multicast* | *vpn4 unicast* | *vpn6 unicast* } **redistribute ospf** *process-id* [**match** { *external* [*1* | *2*] | *internal* | *nssa-external* [*1* | *2*] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
7. Do one of the following:

- **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpn4 unicast** | **vpn6 unicast** } **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpn4 unicast** | **vpn6 unicast** } **redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpn4 unicast** | **vpn6 unicast** } **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpn4 unicast** | **vpn6 unicast** } **network** { *ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*]
- **neighbor** *ip-address* **remote-as** *as-number* **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpn4 unicast** | **vpn6 unicast** }
- **route-policy** *route-policy-name* { **in** | **out** }

8. **commit**

9. **show bgp** [**vrf** *vrf-name*] *ip-address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	route-policy <i>name</i> Example: RP/0/RP0/CPU0: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 { <i>cost-extcommunity-set-name</i> <i>cost-inline-extcommunity-set</i> } [additive] Example: RP/0/RP0/CPU0:router(config)# set extcommunity cost cost_A	Specifies the BGP extended community attribute for cost.
Step 4	end-policy Example: RP/0/RP0/CPU0:router(config)# end-policy	Ends the definition of a route policy and exits route policy configuration mode.
Step 5	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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	Applies the cost community to the attach point (route policy).

	Command or Action	Purpose
	<ul style="list-style-type: none"> • aggregate-address <i>address/mask-length</i> [as-set] [as-confed-set] [summary-only] [route-policy <i>route-policy-name</i>] • address-family { ipv4 unicast ipv4 multicast ipv4 tunnel ipv4 mdt ipv6 unicast ipv6 multicast vpn4 unicast vpn6 unicast } redistribute connected [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] • address-family { ipv4 unicast ipv4 multicast ipv4 tunnel ipv4 mdt ipv6 unicast ipv6 multicast vpn4 unicast vpn6 unicast } redistribute eigrp <i>process-id</i> [match { external internal }] [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] • address-family { ipv4 unicast ipv4 multicast ipv4 tunnel ipv4 mdt ipv6 unicast ipv4 mdt vpn4 unicast vpn6 unicast } redistribute isis <i>process-id</i> [level { 1 1-inter-area 2 }] [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] • address-family { ipv4 unicast ipv4 multicast ipv4 tunnel ipv4 mdt ipv6 unicast ipv6 multicast vpn4 unicast vpn6 unicast } 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>] 	
Step 7	<p>Do one of the following:</p> <ul style="list-style-type: none"> • address-family { ipv4 unicast ipv4 multicast ipv4 tunnel ipv4 mdt ipv6 unicast ipv6 multicast vpn4 unicast vpn6 unicast } 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>] • address-family { ipv4 unicast ipv4 multicast ipv4 tunnel ipv4 mdt ipv6 unicast ipv6 multicast vpn4 unicast vpn6 unicast } redistribute rip [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] • address-family { ipv4 unicast ipv4 multicast ipv4 tunnel ipv4 mdt ipv6 unicast ipv6 multicast vpn4 unicast vpn6 unicast } redistribute static [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] • address-family { ipv4 unicast ipv4 multicast ipv4 tunnel ipv4 mdt ipv6 unicast ipv6 multicast vpn4 unicast vpn6 unicast } 	

	Command or Action	Purpose
	<pre>network { ip-address/prefix-length ip-address mask } [route-policy route-policy-name]</pre> <ul style="list-style-type: none"> • neighbor <i>ip-address</i> remote-as <i>as-number</i> • address-family { ipv4 unicast ipv4 multicast ipv4 tunnel ipv4 mdt ipv6 unicast ipv6 multicast vpn4 unicast vpn6 unicast } • route-policy <i>route-policy-name</i> { in out } 	
Step 8	commit	
Step 9	<pre>show bgp [vrf vrf-name] ip-address</pre> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show bgp 172.168.40.24</pre>	Displays the cost community in the following format: Cost: <i>POI</i> : <i>cost-community-ID</i> : <i>cost-number</i>

Configuring Software to Store Updates from a 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. See the [Resetting Neighbors Using BGP Inbound Soft Reset](#), on page 169.



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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	

	Command or Action	Purpose
Step 2	router bgp <i>as-number</i> Example: <pre>RP/0/RP0/CPU0: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	neighbor <i>ip-address</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</pre>	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: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast</pre>	<p>Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.</p> <p>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</p>
Step 5	soft-reconfiguration inbound [<i>always</i>] Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# soft-reconfiguration inbound always</pre>	<p>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.</p> <p>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.</p>
Step 6	commit	

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.

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)
- Private IPv4 and IPv6 (IPv4/v6 address family inside VRF)

BGP Persistence Configuration: Example

This example sets long lived graceful restart (LLGR) stale-time of 16777215 on BGP neighbor 3.3.3.3.

```
router bgp 100
 neighbor 3.3.3.3
  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
```

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.

No Automatic Shutdown

The Graceful Maintenance feature does not perform any shutdown. When Graceful Maintenance is configured, it remains configured, even through system restarts. It is intended to be used in conjunction with a shutdown of a router or a BGP neighbor. The operator must explicitly shut down whenever it is needed. After Graceful Maintenance is no longer required, the operator must explicitly deactivate it. Graceful Maintenance may be deactivated either after the shutdown is completed, or after the deactivated facilities are again brought up. Whether to leave Graceful Maintenance activated through a bring-up operation depends on whether the transient routing during the bring-up operation is considered a problem.

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: <pre>RP/0/RP0/CPU0: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>RP/0/RP0/CPU0:router(config-bgp)# graceful-maintenance activate all-neighbbors</pre>	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. 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. Use the retain-routes option when only BGP must be brought down instead of the entire router, and when it is

	Command or Action	Purpose
		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.
Step 4	commit	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router(config-bgp-nbr)# graceful-maintenance activate	Announces routes with Graceful Maintenance attributes.

	Command or Action	Purpose
Step 5	commit	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1	Places the router in neighbor group configuration mode.
Step 4	graceful-maintenance activate Example: RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# graceful-maintenance activate	Announces routes with Graceful Maintenance attributes.
Step 5	commit	

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.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **graceful-maintenance as-prepends** *value* | **local-preference** *value*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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 <i>as-number</i> Example: RP/0/RP0/CPU0: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 <i>value</i> local-preference <i>value</i> Example: RP/0/RP0/CPU0: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

	Command or Action	Purpose
		<p>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.</p> <p>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.</p>

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
```

L3VPN iBGP PE-CE

The L3VPN iBGP PE-CE feature helps establish an iBGP (internal Border Gateway Protocol) session between the provider edge (PE) and customer edge (CE) devices to exchange BGP routing information. A BGP session between two BGP peers is said to be an iBGP session if the BGP peers are in the same autonomous systems.

L3VPN iBGP PE-CE Overview

When BGP is used as the provider edge (PE) or the customer edge (CE) routing protocol, the peering sessions are configured as external peering between the VPN provider autonomous system (AS) and the customer network autonomous system. The L3VPN iBGP PE-CE feature enables the PE and CE devices to exchange Border Gateway Protocol (BGP) routing information by peering as internal Border Gateway Protocol (iBGP) instead of the widely-used external BGP peering between the PE and the CE. This mechanism applies at each PE device where a VRF-based CE is configured as iBGP. This eliminates the need for service providers (SPs) to configure autonomous system override for the CE. With this feature enabled, there is no need to configure the virtual private network (VPN) sites using different autonomous systems.

The **neighbor internal-vpn-client** command enables PE devices to make an entire VPN cloud act as an internal VPN client to the CE devices. These CE devices are connected internally to the VPN cloud through the iBGP PE-CE connection inside the VRF. After this connection is established, the PE device encapsulates the CE-learned path into an attribute called ATTR_SET and carries it in the iBGP-sourced path throughout the VPN core to the remote PE device. At the remote PE device, this attribute is assigned with individual attributes and the source CE path is extracted and sent to the remote CE devices.

ATTR_SET is an optional transitive attribute that carries the CE path attributes received. The ATTR_SET attribute is encoded inside the BGP update message as follows:

```

+-----+
| Attr Flags (O|T) Code = 128 |
+-----+
| Attr. Length (1 or 2 octets) |
+-----+
| Origin AS (4 octets) |
+-----+
| Path attributes (variable) |
+-----+

```

Origin AS is the AS of the VPN customer for which the ATTR_SET is generated. The minimum length of ATTR_SET is four bytes and the maximum is the maximum supported for a path attribute after taking into consideration the mandatory fields and attributes in the BGP update message. It is recommended that the maximum length is limited to 3500 bytes. ATTR_SET must not contain the following attributes: MP_REACH, MP_UNREACH, NEW_AS_PATH, NEW_AGGR, NEXT_HOP and ATTR_SET itself (ATTR_SET inside ATTR_SET). If these attributes are found inside the ATTR_SET, the ATTR_SET is considered invalid and the corresponding error handling mechanism is invoked.

Restrictions for L3VPN iBGP PE-CE

The following restrictions apply to configuring L3VPN iBGP PE-CE:

- When the iBGP PE CE feature is toggled and the neighbor no longer supports route-refresh or soft-reconfiguration inbound, a manual session flap must be done to see the change. When this occurs, the following message is displayed:

```

RP/0/0/CPU0: %ROUTING-BGP-5-CFG_CHG_RESET: Internal VPN client configuration change on
neighbor 10.10.10.1 requires HARD reset
(clear bgp 10.10.10.1) to take effect.

```

- iBGP PE CE CLI configuration is not available for peers under default-VRF, except for neighbor/session-group.
- This feature does not work on regular VPN clients (eBGP VPN clients).

- Attributes packed inside the ATTR_SET reflects changes made by the inbound route-policy on the iBGP CE and does not reflect the changes made by the export route-policy for the specified VRF.
- Different VRFs of the same VPN (that is, in different PE routers) that are configured with iBGP PE-CE peering sessions must use different Route Distinguisher (RD) values under respective VRFs. The iBGP PE CE feature does not work if the RD values are the same for the ingress and egress VRF.

Configuring L3VPN iBGP PE-CE

L3VPN iBGP PE-CE can be enabled on the neighbor, neighbor-group, or session-group. To configure L3VPN iBGP PE-CE, follow these steps:

Before you begin

The CE must be an internal BGP peer.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **vrf** *vrf-name*
4. **neighbor** *ip-address* **internal-vpn-client**
5. **commit**
6. **show bgp vrf** *vrf-name* **neighbors** *ip-address*
7. **show bgp** {*vpn4*|*vpn6* } **unicast rd**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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	vrf <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config-bgp)# vrf blue	Configures a VRF instance.
Step 4	neighbor <i>ip-address</i> internal-vpn-client Example: RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 10.0.0.0 internal-vpn-client	Configures a CE neighboring device with which to exchange routing information. The neighbor internal-vpn-client command stacks the iBGP-CE neighbor path in the VPN attribute set.
Step 5	commit	

	Command or Action	Purpose
Step 6	<code>show bgp vrf vrf-name neighbors ip-address</code>	Displays whether the iBGP PE-CE feature is enabled for the VRF CE peer, or not.
Step 7	<code>show bgp { vpv4 vpv6 } unicast rd</code>	Displays the ATTR_SET attributes in the command output when the L3VPN iBGP PE-CE is enabled on a CE.

Example

Example: Configuring L3VPN iBGP PE-CE

The following example shows how to configure L3VPN iBGP PE-CE:

```
R1(config-bgp-vrf-nbr)#neighbor 10.10.10.1 ?
. . .
  internal-vpn-client      Preserve iBGP CE neighbor path in ATTR_SET across VPN core
. . .
R1(config-bgp-vrf-nbr)#neighbor 10.10.10.1 internal-vpn-client
router bgp 65001
  bgp router-id 100.100.100.2
  address-family ipv4 unicast
  address-family vpv4 unicast
  !
  vrf ce-ibgp
    rd 65001:100
    address-family ipv4 unicast
    !
  neighbor 10.10.10.1
    remote-as 65001
    internal-vpn-client
```

The following is an example of the output of the `show bgp vrf vrf-name neighbors ip-address` command when the L3VPN iBGP PE-CE is enabled on a CE peer:

```
R1#show bgp vrf ce-ibgp neighbors 10.10.10.1
BGP neighbor is 10.10.10.1, vrf ce-ibgp
  Remote AS 65001, local AS 65001, internal link
  Remote router ID 100.100.100.1
  BGP state = Established, up for 00:00:19
  . . .
  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
CE attributes will be preserved across the core
  Received 2 messages, 0 notifications, 0 in queue
  Sent 2 messages, 0 notifications, 0 in queue
  . . .
```

The following is an example of the output of the `show bgp vpv4/vpn6 unicast rd` command when the L3VPN iBGP PE-CE is enabled on a CE peer:

```
BGP routing table entry for 1.1.1.0/24, Route Distinguisher: 200:300
Versions:
  Process          bRIB/RIB  SendTblVer
  Speaker          10        10
Last Modified: Aug 28 13:11:17.000 for 00:01:00
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
Local, (Received from a RR-client)
  20.20.20.2 from 20.20.20.2 (100.100.100.2)
  Received Label 24000
  Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
  not-in-vrf Received Path ID 0, Local Path ID 1, version 10
  Extended community: RT:228:237
ATTR-SET [
  Origin-AS: 200
  AS-Path: 51320 52325 59744 12947 21969 50346 18204 36304 41213
23906 33646
  Origin: incomplete
  Metric: 204
  Local-Pref: 234
  Aggregator: 304 34.3.3.3
  Atomic Aggregator
  Community: 1:60042 2:41661 3:47008 4:9280 5:39778 6:1069 7:15918
8:8994 9:52701
10:10268 11:26276 12:8506 13:7131 14:65464 15:14304 16:33615 17:54991
18:40149 19:19401
  Extended community: RT:100:1 RT:1.1.1.1:1]

```

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.

For more information on the commands for flow-tag propagation see the BGP Commands module in the *Routing Command Reference for Cisco CRS Routers*.

Restrictions for flow-tag propagation

Some restrictions are placed with regard to using Quality-of-service Policy Propagation Using Border Gateway Protocol (QPPB). 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-if)# interface GigabitEthernet 0/0/0/0	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 }	Enables flow-tag policy propagation on source or destination IP address on an interface.
	Example: RP/0/RP0/CPU0:router(config-if)# ipv4 bgp policy propagation input flow-tag source	
Step 4	commit	

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
!

RP/0/RSP0/CPU0:CRS-0#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
!

RP/0/RSP0/CPU0:CRS-0#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
!

```

Configuring a VPN Routing and Forwarding Instance in BGP

Layer 3 (virtual private network) VPN can be configured only if there is an available Layer 3 VPN license for the line card slot on which the feature is being configured. If advanced IP license is enabled, 4096 Layer 3 VPN routing and forwarding instances (VRFs) can be configured on an interface. If the infrastructure VRF license is enabled, eight Layer 3 VRFs can be configured on the line card.

See the Software Entitlement on Cisco IOS XR Software module in *System Management Configuration Guide for Cisco CRS Routers* for more information on advanced IP licencing.

The following error message appears if the appropriate licence is not enabled:

```

RP/0/RP0/CPU0:router#LC/0/0/CPU0:Dec 15 17:57:53.653 : rsi_agent[247]:
%LICENSE-CRS_LICENSE-2-INFRA_VRF_NEEDED : 5 VRF(s) are configured without license A9K-iVRF-LIC
in violation of the Software Right To Use Agreement.
This feature may be disabled by the system without the appropriate license.
Contact Cisco to purchase the license immediately to avoid potential service interruption.

```



Note An AIP license is not required for configuring L2VPN services.

The following tasks are used to configure a VPN routing and forwarding (VRF) instance in BGP:

Defining Virtual Routing and Forwarding Tables in Provider Edge Routers

Perform this task to define the VPN routing and forwarding (VRF) tables in the provider edge (PE) routers.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf-name*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **maximum prefix** *maximum* [*threshold*]
5. **import route-policy** *policy-name*
6. **import route-target** [*as-number : nn* | *ip-address : nn*]
7. **export route-policy** *policy-name*
8. **export route-target** [*as-number : nn* | *ip-address : nn*]
9. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	vrf <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config)# vrf vrf_pe	Configures a VRF instance.
Step 3	address-family { ipv4 ipv6 } unicast Example: RP/0/RP0/CPU0:router(config-vrf)# 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	maximum prefix <i>maximum</i> [<i>threshold</i>] Example: RP/0/RP0/CPU0:router(config-vrf-af)# maximum prefix 2300	Configures a limit to the number of prefixes allowed in a VRF table. A maximum number of routes is applicable to dynamic routing protocols as well as static or connected routes. You can specify a threshold percentage of the prefix limit using the <i>mid-threshold</i> argument.
Step 5	import route-policy <i>policy-name</i> Example: RP/0/RP0/CPU0:router(config-vrf-af)# import route-policy policy_a	(Optional) Provides finer control over what gets imported into a VRF. This import filter discards prefixes that do not match the specified <i>policy-name</i> argument.
Step 6	import route-target [<i>as-number : nn</i> <i>ip-address : nn</i>] Example: RP/0/RP0/CPU0:router(config-vrf-af)# import route-target 234:222	Specifies a list of route target (RT) extended communities. Only prefixes that are associated with the specified import route target extended communities are imported into the VRF.

	Command or Action	Purpose
Step 7	export route-policy <i>policy-name</i> Example: RP/0/RP0/CPU0:router(config-vrf-af)# export route-policy policy_b	(Optional) Provides finer control over what gets exported into a VRF. This export filter discards prefixes that do not match the specified <i>policy-name</i> argument.
Step 8	export route-target [<i>as-number : nn</i> <i>ip-address : nn</i>] Example: RP/0/RP0/CPU0:router(config-vrf-af)# export route-target 123;234	Specifies a list of route target extended communities. Export route target communities are associated with prefixes when they are advertised to remote PEs. The remote PEs import them into VRFs which have import RTs that match these exported route target communities.
Step 9	commit	

Configuring the Route Distinguisher

The route distinguisher (RD) makes prefixes unique across multiple VPN routing and forwarding (VRF) instances.

In the L3VPN multipath same route distinguisher (RD) environment, the determination of whether to install a prefix in RIB or not is based on the prefix's bestpath. In a rare misconfiguration situation, where the best path is not a valid path to be installed in RIB, BGP drops the prefix and does not consider the other paths. The behavior is different for different RD setup, where the non-best multipath will be installed if the best multipath is invalid to be installed in RIB.

Perform this task to configure the RD.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp router-id** *ip-address*
4. **vrf** *vrf-name*
5. **rd** { *as-number : nn* | *ip-address : nn* | **auto** }
6. Do one of the following:
 - **end**
 - **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router bgp 120	Enters BGP configuration mode allowing you to configure the BGP routing process.

	Command or Action	Purpose
Step 3	bgp router-id <i>ip-address</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 10.0.0.0</pre>	Configures a fixed router ID for the BGP-speaking router.
Step 4	vrf <i>vrf-name</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_pe</pre>	Configures a VRF instance.
Step 5	rd { <i>as-number : nn</i> <i>ip-address : nn</i> auto } Example: <pre>RP/0/RP0/CPU0:router(config-bgp-vrf)# rd 345:567</pre>	<p>Configures the route distinguisher.</p> <p>Use the auto keyword if you want the router to automatically assign a unique RD to the VRF.</p> <p>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. 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>
Step 6	<p>Do one of the following:</p> <ul style="list-style-type: none"> • end • commit <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf)# end</pre> <p>or</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf)# commit</pre>	<p>Saves configuration changes.</p> <ul style="list-style-type: none"> • When you issue the end command, the system prompts you to commit changes: <pre>Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:</pre> <ul style="list-style-type: none"> • Entering yes saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC configuration mode. • Entering no exits the configuration session and returns the router to EXEC configuration mode without committing the configuration changes. • Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the commit command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring BGP to Advertise VRF Routes for Multicast VPN from PE to PE

Perform these tasks to enable multicast VPN routing for IPv4 and IPv6 address families from one provider edge (PE) router to another:

Advertising VRF Routes for MVPNv4 from PE to PE

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp router-id** *ip-address*
4. **address-family** { **ipv4** | **ipv6** } **unicast**
5. **exit**
6. **address-family** **vpnv4 unicast**
7. **exit**
8. **address-family** **ipv4 mdt**
9. **exit**
10. **neighbor** *ip-address*
11. **remote-as** *as-number*
12. **update-source** *type interface-path-id*
13. **address-family** { **ipv4** | **ipv6** } **unicast**
14. **exit**
15. **address-family** **vpnv4 unicast**
16. **exit**
17. **vrf** *vrf-name*
18. **rd** { *as-number : nn* | *ip-address : nn* | **auto** }
19. **address-family** { **ipv4** | **ipv6** } **unicast**
20. 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*]
21. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	

	Command or Action	Purpose
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router bgp 100	Enters BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	bgp router-id <i>ip-address</i> Example: RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 1.1.1.1	Configures a fixed router ID for a BGP-speaking router.
Step 4	address-family { ipv4 ipv6 } unicast Example: RP/0/RP0/CPU0: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 5	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits IPv4 address family configuration submode and reenters BGP configuration submode.
Step 6	address-family vpnv4 unicast Example: RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv4 unicast	Enters VPNv4 address family configuration submode.
Step 7	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits IPv4 address-family configuration submode and reenters BGP configuration submode.
Step 8	address-family ipv4 mdt Example: RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 mdt	Configures an IPv4 address-family multicast distribution tree (MDT).
Step 9	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.

	Command or Action	Purpose
Step 10	neighbor <i>ip-address</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.16.1.1</pre>	Places the PE router in neighbor configuration submode.
Step 11	remote-as <i>as-number</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 100</pre>	Creates a neighbor and assigns the neighbor a remote autonomous system number, which can be from 1 to 65535.
Step 12	update-source <i>type interface-path-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source loopback 0</pre>	<p>Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.</p> <p>The <i>interface-type interface-id</i> arguments specify the type and ID number of the interface, such as GigabitEthernet or Loopback. Use the CLI help (?) to see a list of all the possible interface types and their ID numbers.</p>
Step 13	address-family { <i>ipv4</i> <i>ipv6</i> } unicast Example: <pre>RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast</pre>	<p>Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.</p> <p>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</p>
Step 14	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# exit</pre>	Exits the neighbor address family configuration submode.
Step 15	address-family vpn4 unicast Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpn4 unicast</pre>	<p>Specifies the address family as VPNv4 and enters address family configuration submode.</p> <p>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</p>
Step 16	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# exit</pre>	Exits BGP neighbor address family configuration submode.
Step 17	vrf <i>vrf-name</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# vrf vpn1</pre>	Enables BGP routing for a particular VRF on the PE router.

	Command or Action	Purpose
Step 18	<p>rd { <i>as-number</i> : <i>nn</i> <i>ip-address</i> : <i>nn</i> auto }</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf)# rd 1:1</pre>	<p>Configures the route distinguisher.</p> <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>
Step 19	<p>address-family { <i>ipv4</i> <i>ipv6</i> } unicast</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast</pre>	<p>Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submenu.</p> <p>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</p>
Step 20	<p>Do one of the following:</p> <ul style="list-style-type: none"> • redistribute connected [<i>metric</i> <i>metric-value</i>] [route-policy <i>route-policy-name</i>] • redistribute eigrp <i>process-id</i> [match { external internal }] [<i>metric</i> <i>metric-value</i>] [route-policy <i>route-policy-name</i>] • redistribute isis <i>process-id</i> [level { 1 1-inter-area 2 }] [<i>metric</i> <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] }] [<i>metric</i> <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] }] [<i>metric</i> <i>metric-value</i>] [route-policy <i>route-policy-name</i>] • redistribute rip [<i>metric</i> <i>metric-value</i>] [route-policy <i>route-policy-name</i>] • redistribute static [<i>metric</i> <i>metric-value</i>] [route-policy <i>route-policy-name</i>] <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf-af)# redistribute ospf 1</pre>	<p>Configures redistribution of a protocol into the VRF address family context.</p>

	Command or Action	Purpose
Step 21	commit	

Advertising VRF Routes for MVPNv6 from PE to PE

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp router-id** *ip-address*
4. **address-family ipv6 unicast**
5. **address-family vpnv6 unicast**
6. **exit**
7. **neighbor-group** *vrf-name*
8. **remote-as** *as-number*
9. **update-source** *interface-type interface-id*
10. **address-family vpnv6 unicast**
11. **exit**
12. **exit**
13. **neighbor** *ip-address*
14. **remote-as** *as-number*
15. **use neighbor-group** *vpn-name*
16. **update-source** *interface-type interface-id*
17. **address-family ipv6 unicast**
18. **exit**
19. **address-family vpnv6 unicast**
20. **exit**
21. **exit**
22. **vrf** *vrf-name*
23. **rd** { *as-number : nn* | *ip-address : nn* | **auto** }
24. **exit**
25. **vrf** *vrf-name*
26. **rd** { *as-number : nn* | *ip-address : nn* | **auto** }
27. **address-family ipv6 unicast**
28. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: RP/0/RP0/CPU0:router# <code>configure</code>	Enters global configuration mode.

	Command or Action	Purpose
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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	bgp router-id <i>ip-address</i> Example: RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 1.1.1.1 Configures a fixed router ID for a BGP-speaking router.	
Step 4	address-family ipv6 unicast Example: RP/0/RP0/CPU0:router(config-bgp)# address-family ipv6 unicast	Specifies the address family as IPv6 and enters IPv6 neighbor address family configuration submenu. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 5	address-family vpn6 unicast Example: RP/0/RP0/CPU0:router(config-bgp)# address-family vpn6 unicast	Enters VPNv6 address family configuration submenu.
Step 6	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the VPNv6 address family configuration submenu.
Step 7	neighbor-group <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config-bgp)# neighbor-group vpn22	Places the router in neighbor group configuration submenu.
Step 8	remote-as <i>as-number</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# remote-as 100	Creates a neighbor and assigns the neighbor a remote autonomous system number, which can be from 1 to 65535.
Step 9	update-source <i>interface-type interface-id</i> Example: RP/0/RP0/CPU0: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.

	Command or Action	Purpose
Step 10	address-family vpnv6 unicast Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family vpnv6 unicast</pre>	<p>Specifies the address family as VPNv6 and enters address family configuration submenu.</p> <p>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</p>
Step 11	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit</pre>	Exits the neighbor group address family configuration submenu.
Step 12	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit</pre>	Exits BGP neighbor group configuration submenu.
Step 13	neighbor ip-address Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# neighbor 1.1.1.2</pre>	Places a PE router in neighbor group configuration submenu.
Step 14	remote-as as-number Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 100</pre>	Creates a neighbor and assigns it a remote autonomous system number, which can be from 1 to 65535.
Step 15	use neighbor-group vpn-name Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group vpn22</pre>	(Optional) Specifies that the BGP neighbor inherits the configuration from the specified VPN neighbor group.
Step 16	update-source interface-type interface-id Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source loopback 0</pre>	<p>Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.</p> <p>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.</p>
Step 17	address-family ipv6 unicast Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv6 unicast</pre>	<p>Specifies the address family as IPv6 and enters IPv6 neighbor address family configuration submenu.</p> <p>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</p>

	Command or Action	Purpose
Step 18	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# exit</pre>	Exits BGP neighbor address family configuration submode.
Step 19	address-family vpnv6 unicast Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpnv6 unicast</pre>	Specifies the address family as VPNv6 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 20	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# exit</pre>	Exits the neighbor address family configuration submode.
Step 21	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# exit</pre>	Exits the BGP neighbor configuration submode.
Step 22	vrf vrf-name Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# vrf vpn1</pre>	Enters BGP VRF configuration submode.
Step 23	rd { as-number : nn ip-address : nn auto } Example: <pre>RP/0/RP0/CPU0:router(config-bgp-vrf)# rd 111:1</pre>	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>
Step 24	exit Example: <pre>RP/0/RP0/CPU0:router(config-bgp-vrf)# exit</pre>	Exits BGP VRF configuration submode.

	Command or Action	Purpose
Step 25	vrf <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# vrf vpn1	Enables BGP routing for a particular VRF on the PE router.
Step 26	rd { <i>as-number : nn</i> <i>ip-address : nn</i> auto } Example: RP/0/RP0/CPU0:router(config-bgp-vrf)# rd 1:1	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>
Step 27	address-family ipv6 unicast Example: RP/0/RP0/CPU0:router(config-bgp-vrf)# address-family ipv6 unicast	Specifies the address family as IPv6 and enters IPv6 VRF address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 28	commit	

Configuring PE-PE or PE-RR Interior BGP Sessions

To enable BGP to carry VPN reachability information between provider edge (PE) routers you must configure the PE-PE interior BGP (iBGP) sessions. A PE uses VPN information carried from the remote PE router to determine VPN connectivity and the label value to be used so the remote (egress) router can demultiplex the packet to the correct VPN during packet forwarding.

The PE-PE, PE-route reflector (RR) iBGP sessions are defined to all PE and RR routers that participate in the VPNs configured in the PE router.

Perform this task to configure PE-PE iBGP sessions and to configure global VPN options on a PE.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { *vpn4 unicast* | *vpn6 unicast* }
4. **exit**
5. **neighbor** *ip-address*
6. **remote-as** *as-number*

7. **description** *text*
8. **password** { **clear** | **encrypted** } *password*
9. **shutdown**
10. **timers** *keepalive hold-time*
11. **update-source** *type interface-id*
12. **address-family** { **vpn4 unicast** | **vpn6 unicast** }
13. **route-policy** *route-policy-name* **in**
14. **route-policy** *route-policy-name* **out**
15. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 { vpn4 unicast vpn6 unicast } Example: RP/0/RP0/CPU0:router(config-bgp)# address-family vpn4 unicast	Enters VPN address family configuration mode.
Step 4	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.
Step 5	neighbor <i>ip-address</i> Example: RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.16.1.1	Configures a PE iBGP neighbor.
Step 6	remote-as <i>as-number</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1	Assigns the neighbor a remote autonomous system number.
Step 7	description <i>text</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# description neighbor 172.16.1.1	(Optional) Provides a description of the neighbor. The description is used to save comments and does not affect software function.

	Command or Action	Purpose
Step 8	password { clear encrypted } <i>password</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# password encrypted 123abc</pre>	Enables Message Digest 5 (MD5) authentication on the TCP connection between the two BGP neighbors.
Step 9	shutdown Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# shutdown</pre>	Terminates any active sessions for the specified neighbor and removes all associated routing information.
Step 10	timers <i>keepalive hold-time</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# timers 12000 200</pre>	Set the timers for the BGP neighbor.
Step 11	update-source <i>type interface-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source gigabitEthernet 0/1/5/0</pre>	Allows iBGP sessions to use the primary IP address from a specific interface as the local address when forming an iBGP session with a neighbor.
Step 12	address-family { <i>vpn4 unicast</i> <i>vpn6 unicast</i> } Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpn4 unicast</pre>	Enters VPN neighbor address family configuration mode.
Step 13	route-policy <i>route-policy-name</i> in Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pe-pe-vpn-in in</pre>	Specifies a routing policy for an inbound route. The policy can be used to filter routes or modify route attributes.
Step 14	route-policy <i>route-policy-name</i> out Example: <pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pe-pe-vpn-out out</pre>	Specifies a routing policy for an outbound route. The policy can be used to filter routes or modify route attributes.
Step 15	commit	

Configuring Route Reflector to Hold Routes That Have a Defined Set of RT Communities

A provider edge (PE) needs to hold the routes that match the import route targets (RTs) of the VPNs configured on it. The PE router can discard all other VPNv4 (Cisco XR 12000 Series Router and Cisco CRS-1) and VPNv6 (Cisco XR 12000 Series Router only) routes. But, a route reflector (RR) must retain all VPNv4 and VPNv6 routes, because it might peer with PE routers and different PEs might require different RT-tagged

VPNv4 and VPNv6 routes (making RRs non-scalable). You can configure an RR to only hold routes that have a defined set of RT communities. Also, a number of the RRs can be configured to service a different set of VPNs (thereby achieving some scalability). A PE is then made to peer with all RRs that service the VRFs configured on the PE. When a new VRF is configured with an RT for which the PE does not already hold routes, the PE issues route refreshes to the RRs and retrieves the relevant VPN routes.



Note Note that this process can be more efficient if the PE-RR session supports extended community outbound route filter (ORF).

Perform this task to configure a reflector to retain routes tagged with specific RTs.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **vpn4 unicast** | **vpn6 unicast** }
4. **retain route-target** { **all** | **route-policy** *route-policy-name* }
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: <pre>RP/0/RP0/CPU0: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	address-family { vpn4 unicast vpn6 unicast } Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# address-family vpn4 unicast</pre>	Enters VPN address family configuration mode.
Step 4	retain route-target { all route-policy <i>route-policy-name</i> } Example: <pre>RP/0/RP0/CPU0:router(config-bgp-af)# retain route-target route-policy rr_ext-comm</pre>	Configures a reflector to retain routes tagged with particular RTs. Use the <i>route-policy-name</i> argument for the policy name that lists the extended communities that a path should have in order for the RR to retain that path. Note The all keyword is not required, because this is the default behavior of a route reflector.
Step 5	commit	

Configuring BGP as a PE-CE Protocol

Perform this task to configure BGP on the PE and establish PE-CE communication using BGP. This task can be performed in both VRF and non-VRF configuration.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **vrf** *vrf-name*
4. **bgp router-id** *ip-address*
5. **label mode** { **per-ce** | **per-vrf** }
6. **address-family** { **ipv4** | **ipv6** } **unicast**
7. **network** { *ip-address / prefix-length* | *ip-address mask* }
8. **aggregate-address** *address / mask-length*
9. **exit**
10. **neighbor** *ip-address*
11. **remote-as** *as-number*
12. **password** { **clear** | **encrypted** } *password*
13. **ebgp-multihop** [*ttl-value*]
14. Do one of the following:
 - **address-family** { **ipv4** | **ipv6** } **unicast**
 - **address-family** { **ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast** }
15. **site-of-origin** [*as-number : nn* | *ip-address : nn*]
16. **as-override**
17. **allowas-in** [*as-occurrence-number*]
18. **route-policy** *route-policy-name* **in**
19. **route-policy** *route-policy-name* **out**
20. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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	vrf <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_pe_2	Enables BGP routing for a particular VRF on the PE router.
Step 4	bgp router-id <i>ip-address</i> Example: RP/0/RP0/CPU0:router(config-bgp-vrf)# bgp router-id 172.16.9.9	Configures a fixed router ID for a BGP-speaking router.
Step 5	label mode { per-ce per-vrf }	Configures the MPLS/VPN label mode.

	Command or Action	Purpose
	<p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf)# label mode per-ce</pre>	<ul style="list-style-type: none"> The per-ce keyword configures the per-CE label mode to avoid an extra lookup on the PE router and conserve label space (per-prefix is the default label mode). In this mode, the PE router allocates one label for every immediate next-hop (in most cases, this would be a CE router). This label is directly mapped to the next hop, so there is no VRF route lookup performed during data forwarding. However, the number of labels allocated would be one for each CE rather than one for each VRF. Because BGP knows all the next hops, it assigns a label for each next hop (not for each PE-CE interface). When the outgoing interface is a multiaccess interface and the media access control (MAC) address of the neighbor is not known, Address Resolution Protocol (ARP) is triggered during packet forwarding. The per-vrf keyword configures the same label to be used for all the routes advertised from a unique VRF.
Step 6	<p>address-family { ipv4 ipv6 } unicast</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast</pre>	<p>Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submenu.</p> <p>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</p>
Step 7	<p>network { <i>ip-address / prefix-length</i> <i>ip-address mask</i> }</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf-af)# network 172.16.5.5</pre>	<p>Originates a network prefix in the address family table in the VRF context.</p>
Step 8	<p>aggregate-address <i>address / mask-length</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf-af)# aggregate-address 10.0.0.0/24</pre>	<p>Configures aggregation in the VRF address family context to summarize routing information to reduce the state maintained in the core. This summarization introduces some inefficiency in the PE edge, because an additional lookup is required to determine the ultimate next hop for a packet. When configured, a summary prefix is advertised instead of a set of component prefixes, which are more specific of the aggregate. The PE advertises only one label for the aggregate. Because component prefixes could have different next hops to CEs, an additional lookup has to be performed during data forwarding.</p>
Step 9	<p>exit</p> <p>Example:</p>	<p>Exits the current configuration mode.</p>

	Command or Action	Purpose
	<pre>RP/0/RP0/CPU0:router(config-bgp-vrf-af)# exit</pre>	
Step 10	<p>neighbor <i>ip-address</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 10.0.0.0</pre>	Configures a CE neighbor. The <i>ip-address</i> argument must be a private address.
Step 11	<p>remote-as <i>as-number</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# remote-as 2</pre>	Configures the remote AS for the CE neighbor.
Step 12	<p>password { clear encrypted } <i>password</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# password encrypted 234xyz</pre>	Enable Message Digest 5 (MD5) authentication on a TCP connection between two BGP neighbors.
Step 13	<p>ebgp-multihop [<i>ttl-value</i>]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop 55</pre>	Configures the CE neighbor to accept and attempt BGP connections to external peers residing on networks that are not directly connected.
Step 14	<p>Do one of the following:</p> <ul style="list-style-type: none"> • address-family { ipv4 ipv6 } unicast • address-family { ipv4 { unicast labeled-unicast } ipv6 unicast } <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast</pre>	<p>Specifies either an IPv4 (unicast or labeled-unicast) or IPv6 unicast address family and enters address family configuration submode.</p> <p>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</p>
Step 15	<p>site-of-origin [<i>as-number : nn</i> <i>ip-address : nn</i>]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# site-of-origin 234:111</pre>	Configures the site-of-origin (SoO) extended community. Routes that are learned from this CE neighbor are tagged with the SoO extended community before being advertised to the rest of the PEs. SoO is frequently used to detect loops when as-override is configured on the PE router. If the prefix is looped back to the same site, the PE detects this and does not send the update to the CE.
Step 16	<p>as-override</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# as-override</pre>	<p>Configures AS override on the PE router. This causes the PE router to replace the CE's ASN with its own (PE) ASN.</p> <p>Note This loss of information could lead to routing loops; to avoid loops caused by as-override, use it in conjunction with site-of-origin.</p>

	Command or Action	Purpose
Step 17	allowas-in [<i>as-occurrence-number</i>] Example: RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# allowas-in 5	Allows an AS path with the PE autonomous system number (ASN) a specified number of times. Hub and spoke VPN networks need the looping back of routing information to the HUB PE through the HUB CE. When this happens, due to the presence of the PE ASN, the looped-back information is dropped by the HUB PE. To avoid this, use the allowas-in command to allow prefixes even if they have the PEs ASN up to the specified number of times.
Step 18	route-policy <i>route-policy-name</i> in Example: RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pe_ce_in_policy in	Specifies a routing policy for an inbound route. The policy can be used to filter routes or modify route attributes.
Step 19	route-policy <i>route-policy-name</i> out Example: RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pe_ce_out_policy out	Specifies a routing policy for an outbound route. The policy can be used to filter routes or modify route attributes.
Step 20	commit	

Redistribution of IGP to BGP

Perform this task to configure redistribution of a protocol into the VRF address family.

Even if Interior Gateway Protocols (IGPs) are used as the PE-CE protocol, the import logic happens through BGP. Therefore, all IGP routes have to be imported into the BGP VRF table.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **vrf** *vrf-name*
4. **address-family** { *ipv4* | *ipv6* } **unicast**
5. 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*]

6. commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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	vrf <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_a	Enables BGP routing for a particular VRF on the PE router.
Step 4	address-family { ipv4 ipv6 } unicast Example: RP/0/RP0/CPU0: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 5	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>] • redistribute static [metric <i>metric-value</i>] [route-policy <i>route-policy-name</i>] Example: RP/0/RP0/CPU0:router(config-bgp-vrf-af)# redistribute eigrp 23	Configures redistribution of a protocol into the VRF address family context. The redistribute command is used if BGP is not used between the PE-CE routers. If BGP is used between PE-CE routers, the IGP that is used has to be redistributed into BGP to establish VPN connectivity with other PE sites. Redistribution is also required for inter-table import and export.

	Command or Action	Purpose
Step 6	commit	

Configuring 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.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **keychain** *name*
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 <i>ip-address</i> Example: RP/0/RP0/CPU0: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 <i>as-number</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002	Creates a neighbor and assigns a remote autonomous system number to it.
Step 5	keychain <i>name</i> Example:	Configures keychain-based authentication.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-bgp-nbr)# keychain kych_a	
Step 6	commit	

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. commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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 4	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.
Step 5	address-family { vpnv4 vpnv6 } unicast Example: RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv4 unicast	Specifies the address family and enters the address family configuration submode. 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: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.
Step 7	address-family ipv4 mdt Example: RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 mdt	Specifies the multicast distribution tree (MDT) address family.
Step 8	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.

	Command or Action	Purpose
Step 9	neighbor <i>ip-address</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source loopback 0	<p>Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.</p> <p>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.</p>
Step 12	address-family { <i>ipv4</i> <i>ipv6</i> } unicast Example: RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast	<p>Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.</p> <p>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</p>
Step 13	exit Example: RP/0/RP0/CPU0:router(config-bgp-nbr-af)# exit	(Optional) Exits the current configuration mode.
Step 14	address-family { <i>vpn4</i> <i>vpn6</i> } unicast Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpn4 unicast	<p>(Optional) Enters address family configuration submode for the specified address family.</p> <p>Note Required if you are configuring multicast MVPN. If configuring MVPNv6, use the vpn6 keyword.</p>
Step 15	exit Example: RP/0/RP0/CPU0:router(config-bgp-nbr-af)# exit	Exits the current configuration mode.
Step 16	address-family ipv4 mdt Example: RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 mdt	Specifies the multicast distribution tree (MDT) address family.

	Command or Action	Purpose
Step 17	exit Example: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.
Step 18	vrf vrf-name Example: RP/0/RP0/CPU0:router(config-bgp)# vrf vpn1	(Optional) Enables BGP routing for a particular VRF on the PE router. Note Required if you are configuring multicast MVPN.
Step 19	rd { as-number:nn ip-address:nn auto } Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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 21	Do one of the following: <ul style="list-style-type: none"> • 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] }] [(Optional) Configures redistribution of a protocol into the VRF address family context. Note Required if you are configuring multicast MVPN.

	Command or Action	Purpose
	<pre>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]</pre> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp-vrf-af)# redistribute eigrp 23</pre>	
Step 22	commit	

Disabling a 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	<pre>router bgp as-number</pre> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config)# router bgp 127</pre>	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	<pre>neighbor ip-address</pre> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</pre>	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	<pre>shutdown</pre> <p>Example:</p>	Disables all active sessions for the specified neighbor.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-bgp-nbr)# shutdown	
Step 5	commit	

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.

SUMMARY STEPS

1. **show bgp neighbors**
2. **clear bgp** { ipv4 { unicast | multicast | labeled-unicast | all | tunnel | mdt } | ipv6 { unicast | multicast | 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 } soft [in [prefix-filter] | out]

DETAILED STEPS

	Command or Action	Purpose
Step 1	show bgp neighbors Example: RP/0/RP0/CPU0:router# show bgp neighbors	Verifies that received route refresh capability from the neighbor is enabled.
Step 2	clear bgp { ipv4 { unicast multicast labeled-unicast all tunnel mdt } ipv6 { unicast multicast 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 } soft [in [prefix-filter] out] Example: RP/0/RP0/CPU0:router# clear bgp ipv4 unicast 10.0.0.1 soft in	Soft resets a BGP neighbor. <ul style="list-style-type: none"> • The <i>*</i> keyword resets all BGP neighbors. • The <i>ip-address</i> argument specifies the address of the neighbor to be reset. • The <i>as-number</i> 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.

SUMMARY STEPS

1. **show bgp neighbors**
2. **clear bgp** { ipv4 { unicast | multicast | labeled-unicast | all | tunnel | } | ipv6 { unicast | multicast | 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 } **clear bgp** { ipv4 | ipv6 } { unicast | labeled-unicast } **soft out**

DETAILED STEPS

	Command or Action	Purpose
Step 1	show bgp neighbors Example: RP/0/RP0/CPU0:router# show bgp neighbors	Verifies that received route refresh capability from the neighbor is enabled.
Step 2	clear bgp { ipv4 { unicast multicast labeled-unicast all tunnel } ipv6 { unicast multicast 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 } clear bgp { ipv4 ipv6 } { unicast labeled-unicast } soft out Example: RP/0/RP0/CPU0:router# clear bgp ipv4 unicast 10.0.0.2 soft out	Soft resets a BGP neighbor. <ul style="list-style-type: none"> • The <i>*</i> keyword resets all BGP neighbors. • The <i>ip-address</i> argument specifies the address of the neighbor to be reset. • The <i>as-number</i> 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 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.

SUMMARY STEPS

1. `clear bgp { ipv4 { unicast | multicast | labeled-unicast | all | tunnel | mdt } | ipv6 { unicast | multicast | 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 }`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<pre>clear bgp { ipv4 { unicast multicast labeled-unicast all tunnel mdt } ipv6 { unicast multicast 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 }</pre> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# clear bgp ipv4 unicast 10.0.0.3 graceful soft out</pre>	<p>Clears a BGP neighbor.</p> <ul style="list-style-type: none"> • The <code>*</code> keyword resets all BGP neighbors. • The <code>ip-address</code> argument specifies the address of the neighbor to be reset. • The <code>as-number</code> argument specifies that all neighbors that match the autonomous system number be reset. • The <code>external</code> keyword specifies that all external neighbors are reset. <p>The <code>graceful</code> keyword specifies a graceful restart.</p>

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.

SUMMARY STEPS

1. `clear bgp { ipv4 { unicast | multicast | labeled-unicast | all | tunnel | mdt } | ipv6 { unicast | multicast | 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`
2. `clear bgp external`
3. `clear bgp *`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<pre>clear bgp { ipv4 { unicast multicast labeled-unicast all tunnel mdt } ipv6 { unicast multicast all labeled-unicast } all { unicast multicast all labeled-unicast mdt tunnel } vpnv4 unicast</pre>	Clears a specified neighbor.

	Command or Action	Purpose
	<pre> vrf { vrf-name all } { ipv4 { unicast labeled-unicast } ipv6 unicast } vpnv6 unicast } ip-address</pre> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# clear bgp ipv4 172.20.1.1</pre>	
Step 2	<p>clear bgp external</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# clear bgp external</pre>	Clears all external peers.
Step 3	<p>clear bgp *</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# clear bgp *</pre>	Clears all BGP neighbors.

Displaying 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

	Command or Action	Purpose
Step 1	<p>show bgp cidr-only</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show bgp cidr-only</pre>	Displays routes with nonnatural network masks (classless interdomain routing [CIDR]) routes.

	Command or Action	Purpose
Step 2	show bgp community <i>community-list</i> [exact-match] Example: <pre>RP/0/RP0/CPU0:router# show bgp community 1081:5 exact-match</pre>	Displays routes that match the specified BGP community.
Step 3	show bgp regexp <i>regular-expression</i> Example: <pre>RP/0/RP0/CPU0:router# show bgp regexp "^3 "</pre>	Displays routes that match the specified autonomous system path regular expression.
Step 4	show bgp Example: <pre>RP/0/RP0/CPU0:router# show bgp</pre>	Displays entries in the BGP routing table.
Step 5	show bgp neighbors <i>ip-address</i> [advertised-routes dampened-routes flap-statistics performance-statistics received <i>prefix-filter</i> routes] Example: <pre>RP/0/RP0/CPU0:router# show bgp neighbors 10.0.101.1</pre>	Displays information about the BGP connection to the specified neighbor. <ul style="list-style-type: none"> • 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 <i>prefix-filter</i> keyword and argument display the received prefix list filter. • The routes keyword displays routes learned from the neighbor.
Step 6	show bgp paths Example: <pre>RP/0/RP0/CPU0:router# show bgp paths</pre>	Displays all BGP paths in the database.
Step 7	show bgp neighbor-group <i>group-name</i> configuration Example: <pre>RP/0/RP0/CPU0:router# show bgp neighbor-group group_1 configuration</pre>	Displays the effective configuration for a specified neighbor group, including any configuration inherited by this neighbor group.

	Command or Action	Purpose
Step 8	show bgp summary Example: RP/0/RP0/CPU0:router# show bgp summary	Displays the status of all BGP connections.

Displaying BGP Process Information

Perform this task to display specific BGP process information.

SUMMARY STEPS

1. **show bgp process**
2. **show bgp ipv4 unicast summary**
3. **show bgp vpnv4 unicast summary**
4. **show bgp vrf (vrf-name | all)**
5. **show bgp process detail**
6. **show bgp summary**
7. **show placement program bgp**
8. **show placement program brib**

DETAILED STEPS

	Command or Action	Purpose
Step 1	show bgp process Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router# show bgp vrf vrf_A	Displays BGP VPN virtual routing and forwarding (VRF) information.

	Command or Action	Purpose
Step 5	show bgp process detail Example: <pre>RP/0/RP0/CPU0:router# show bgp processes detail</pre>	Displays detailed process information including the memory used by each of various internal structure types.
Step 6	show bgp summary Example: <pre>RP/0/RP0/CPU0:router# show bgp summary</pre>	Displays the status of all BGP connections.
Step 7	show placement program bgp Example: <pre>RP/0/RP0/CPU0:router# show placement program bgp</pre>	Displays BGP program information. <ul style="list-style-type: none"> • 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: <pre>RP/0/RP0/CPU0:router# show placement program brib</pre>	Displays bRIB program information. <ul style="list-style-type: none"> • 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.

Monitoring BGP Update Groups

This task displays information related to the processing of BGP update groups.

SUMMARY STEPS

1. **show bgp** [ipv4 { unicast | multicast | labeled-unicast | all | tunnel | } | ipv6 { unicast | all | labeled-unicast } | all { unicast | multicast | all | mdt | labeled-unicast | tunnel } | vpnv4 unicast | vrf { vrf-name | all } [ipv4 { unicast | labeled-unicast } | ipv6 unicast] | vpnv6 unicast] update-group [neighbor ip-address | process-id.index [summary | performance-statistics]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	<p>show bgp [ipv4 { unicast multicast labeled-unicast all tunnel }] ipv6 { unicast all labeled-unicast } all { unicast multicast all mdt labeled-unicast tunnel } vpn4 unicast vrf { vrf-name all } [ipv4 { unicast labeled-unicast } ipv6 unicast] vpn6 unicast] update-group [neighbor <i>ip-address</i> <i>process-id.index</i> [summary performance-statistics]]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show bgp update-group 0.0</pre>	<p>Displays information about BGP update groups.</p> <ul style="list-style-type: none"> • The <i>ip-address</i> argument displays the update groups to which that neighbor belongs. • The <i>process-id.index</i> argument selects a particular update group to display and is specified as follows: process ID (dot) index. Process ID range is from 0 to 254. Index range is from 0 to 4294967295. • The summary keyword displays summary information for neighbors in a particular update group. • If no argument is specified, this command displays information for all update groups (for the specified address family). • The performance-statistics keyword displays performance statistics for an update group.

Configuring BGP Nonstop Routing

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **nsr**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	<p>router bgp <i>as-number</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config)# router bgp 120</pre>	Specifies the BGP AS number, and enters the BGP configuration mode, for configuring BGP routing processes.
Step 3	<p>nsr</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-bgp)# nsr</pre>	Activates BGP Nonstop routing.
Step 4	commit	

Configuring Best-External Path Advertisement

Perform the following tasks to advertise the best-external path to the iBGP and route-reflector peers:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. Do one of the following
 - **address-family** { **vpn4 unicast** | **vpn6 unicast** }
 - **vrfvrf-name**{**ipv4 unicast**|**ipv6 unicast**}
4. **advertise best-external**
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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	Do one of the following <ul style="list-style-type: none"> • address-family { vpn4 unicast vpn6 unicast } • vrfvrf-name{ipv4 unicast ipv6 unicast} Example: RP/0/RP0/CPU0:router(config-bgp)# address-family vpn4 unicast	Specifies the address family or VRF address family and enters the address family or VRF address family configuration submenu.
Step 4	advertise best-external Example: RP/0/RP0/CPU0:router(config-bgp-af)# advertise best-external	Advertise the best-external path to the iBGP and route-reflector peers.
Step 5	commit	

Installing Primary Backup Path for Prefix Independent Convergence (PIC)

Perform the following tasks to install a backup path into the forwarding table and provide prefix independent convergence (PIC) in case of a PE-CE link failure:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. Do one of the following
 - **address-family** {*vpn4 unicast* | *vpn6 unicast*}
 - **vrf** *vrf-name* {*ipv4 unicast* | *ipv6 unicast*}
4. **additional-paths selection route-policy** *route-policy-name*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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	Do one of the following <ul style="list-style-type: none"> • address-family {<i>vpn4 unicast</i> <i>vpn6 unicast</i>} • vrf <i>vrf-name</i> {<i>ipv4 unicast</i> <i>ipv6 unicast</i>} Example: RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv4 unicast	Specifies the address family or VRF address family and enters the address family or VRF address family configuration submode.
Step 4	additional-paths selection route-policy <i>route-policy-name</i> Example: RP/0/RP0/CPU0:router(config-bgp-af)# additional-paths selection route-policy ap1	Configures additional paths selection mode for a prefix. Note Use the additional-paths selection command with an appropriate route-policy to calculate backup paths and to enable Prefix Independent Convergence (PIC) functionality. The route-policy configuration is a pre-requisite for configuring the additional-paths selection mode for a prefix. This is an example route-policy configuration to use with additional-selection command: <pre>route-policy ap1 set path-selection backup 1 install end-policy</pre>
Step 5	commit	

Retaining Allocated Local Label for Primary Path

Perform the following tasks to retain the previously allocated local label for the primary path on the primary PE for some configurable time after reconvergence:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **vpn4 unicast** | **vpn6 unicast** }
4. **retain local-label** *minutes*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 { vpn4 unicast vpn6 unicast } Example: RP/0/RP0/CPU0:router(config-bgp)# address-family vpn4 unicast	Specifies the address family and enters the address family configuration submenu.
Step 4	retain local-label <i>minutes</i> Example: RP/0/RP0/CPU0:router(config-bgp-af)# retain local-label 10	Retains the previously allocated local label for the primary path on the primary PE for 10 minutes after reconvergence.
Step 5	commit	

Configuring BGP Additional Paths

Perform these tasks to configure BGP Additional Paths capability:

SUMMARY STEPS

1. **configure**
2. **route-policy** *route-policy-name*
3. **if** *conditional-expression* **then** *action-statement* **else**
4. **pass endif**
5. **end-policy**
6. **router bgp** *as-number*

7. **address-family** {ipv4 {unicast | multicast} | ipv6 {unicast | multicast | l2vpn vpls-vpws | vpnv4 unicast | vpnv6 unicast } }
8. **additional-paths receive**
9. **additional-paths send**
10. **additional-paths selection route-policy** *route-policy-name*
11. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	route-policy <i>route-policy-name</i> Example: RP/0/RP0/CPU0:router (config)#route-policy add_path_policy	Defines the route policy and enters route-policy configuration mode.
Step 3	if conditional-expression then action-statement else Example: RP/0/RP0/CPU0:router (config-rpl)#if community matches-any (*) then set path-selection all advertise else	Decides the actions and dispositions for the given route.
Step 4	pass endif Example: RP/0/RP0/CPU0:router (config-rpl-else) #pass RP/0/RP0/CPU0:router (config-rpl-else) #endif	Passes the route for processing and ends the if statement.
Step 5	end-policy Example: RP/0/RP0/CPU0:router (config-rpl) #end-policy	Ends the route policy definition of the route policy and exits route-policy configuration mode.
Step 6	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 7	address-family {ipv4 {unicast multicast} ipv6 {unicast multicast l2vpn vpls-vpws vpnv4 unicast vpnv6 unicast } } Example: RP/0/RP0/CPU0:router (config-bgp) #address-family ipv4 unicast	Specifies the address family and enters address family configuration submode.
Step 8	additional-paths receive Example: RP/0/RP0/CPU0:router (config-bgp-af) #additional-paths receive	Configures receive capability of multiple paths for a prefix to the capable peers.

	Command or Action	Purpose
Step 9	additional-paths send Example: RP/0/RP0/CPU0:router(config-bgp-af)#additional-paths send	Configures send capability of multiple paths for a prefix to the capable peers .
Step 10	additional-paths selection route-policy <i>route-policy-name</i> Example: RP/0/RP0/CPU0:router(config-bgp-af)#additional-paths selection route-policy add_path_policy	Configures additional paths selection capability for a prefix.
Step 11	commit	

Configuring 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 { <i>ipv4 ipv6</i> } { <i>unicast multicast</i> } Example: RP/0/RP0/CPU0: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 <i>number</i> Example: RP/0/RP0/CPU0:router(config-bgp-af)# maximum-paths ibgp 30	Configures the maximum number of iBGP paths for load sharing.
Step 5	commit	

Originating 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.

SUMMARY STEPS

1. **configure**
2. **route-policy** *aigp_policy*
3. **set aigp-metric***igp-cost*
4. **exit**
5. **router bgp** *as-number*
6. **address-family** {*ipv4* | *ipv6*} **unicast**
7. **redistribute ospf** *osp* **route-policy** *plcy_name***metric** *value*
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	route-policy <i>aigp_policy</i> Example: RP/0/RP0/CPU0:router(config)# route-policy aigp_policy	Enters route-policy configuration mode and sets the route-policy
Step 3	set aigp-metric <i>igp-cost</i> Example: RP/0/RP0/CPU0:router(config-rpl)# set aigp-metric igp-cost	Sets the internal routing protocol cost as the aigp metric.
Step 4	exit Example: RP/0/RP0/CPU0:router(config-rpl)# exit	Exits route-policy configuration mode.

	Command or Action	Purpose
Step 5	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 { <i>ipv4</i> <i>ipv6</i> } unicast Example: RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast	Specifies either the IPv4 or IPv6 address family and enters address family configuration submenu.
Step 7	redistribute ospf osp route-policy <i>plcy_nametric value</i> Example: RP/0/RP0/CPU0:router(config-bgp-af)# redistribute ospf osp route-policy aigp_policy metric 1	Allows the redistribution of AiBGP metric into OSPF.
Step 8	commit	

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

	Command or Action	Purpose
Step 1	configure	
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:	Assigns a remote autonomous system number to the neighbor.

	Command or Action	Purpose
	<code>Router(config-bgp-nbr)#remote-as 100</code>	
Step 5	update-source <i>type interface-path-id</i> Example: <code>Router(config-bgp-nbr)#update-source Loopback0</code>	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 vpn6 unicast</i> } Example: <code>Router(config-bgp-nbr)#address-family vpn6 unicast</code>	Specifies the address family as VPNv4 or VPNv6 and enters neighbor address family configuration mode.
Step 7	accept-own [inheritance-disable] Example: <code>Router(config-bgp-nbr-af)#accept-own</code>	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.

Configuring BGP Link-State

Configuring BGP Link-state

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

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **address-family link-state link-state**
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: <code>RP/0/RP0/CPU0:router(config)# router bgp 100</code>	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: <code>RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.2</code>	Configures a CE neighbor. The ip-address argument must be a private address.

	Command or Action	Purpose
Step 4	remote-as <i>as-number</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1	Configures the remote AS for the CE neighbor.
Step 5	address-family link-state link-state Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family link-state link-state	Distributes BGP link-state information to the specified neighbor.
Step 6	commit	

Configuring Domain Distinguisher

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

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family link-state link-state**
4. **domain-distinguisher** *unique-id*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router(config-bgp)# address-family link-state link-state	Enters address-family link-state configuration mode.

	Command or Action	Purpose
Step 4	domain-distinguisher <i>unique-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-bgp-af) # domain-distinguisher 1234</pre>	Configures unique identifier four-octet ASN. Range is from 1 to 4294967295.
Step 5	commit	

Configuring BGP Permanent Network

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.

SUMMARY STEPS

1. **configure**
2. **prefix-set** *prefix-set-name*
3. **exit**
4. **route-policy** *route-policy-name*
5. **end-policy**
6. **router bgp** *as-number*
7. **address-family** { **ipv4** | **ipv6** } **unicast**
8. **permanent-network** **route-policy** *route-policy-name*
9. **commit**
10. **show bgp** {**ipv4** | **ipv6**} **unicast** *prefix-set*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	prefix-set <i>prefix-set-name</i> Example: <pre>RP/0/RP0/CPU0:router(config)# prefix-set PERMANENT-NETWORK-IPv4 RP/0/RP0/CPU0:router(config-pfx)# 1.1.1.1/32, RP/0/RP0/CPU0:router(config-pfx)# 2.2.2.2/32, RP/0/RP0/CPU0:router(config-pfx)# 3.3.3.3/32 RP/0/RP0/CPU0:router(config-pfx)# end-set</pre>	Enters prefix set configuration mode and defines a prefix set for contiguous and non-contiguous set of bits.
Step 3	exit Example:	Exits prefix set configuration mode and enters global configuration mode.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router (config-pfx) # exit	
Step 4	route-policy <i>route-policy-name</i> Example: RP/0/RP0/CPU0:router (config) # route-policy POLICY-PERMANENT-NETWORK-IPv4 RP/0/RP0/CPU0:router (config-rpl) # if destination in PERMANENT-NETWORK-IPv4 then RP/0/RP0/CPU0:router (config-rpl) # pass RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router (config-rpl) # end-policy	Ends the definition of a route policy and exits route policy configuration mode.
Step 6	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0:router (config) # router bgp 100	Specifies the autonomous system number and enters the BGP configuration mode.
Step 7	address-family { <i>ipv4</i> <i>ipv6</i> } unicast Example: RP/0/RP0/CPU0: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 <i>route-policy-name</i> Example: RP/0/RP0/CPU0: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	commit	
Step 10	show bgp { <i>ipv4</i> <i>ipv6</i> } unicast <i>prefix-set</i> Example: RP/0/RP0/CPU0:router 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.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **address-family** { *ipv4* | *ipv6* } **unicast**
6. **advertise permanent-network**
7. **commit**
8. **show bgp** {*ipv4* | *ipv6*} **unicast neighbor** *ip-address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router bgp 100	Specifies the autonomous system number and enters the BGP configuration mode.
Step 3	neighbor <i>ip-address</i> Example: RP/0/RP0/CPU0: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 <i>as-number</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 4713	Assigns the neighbor a remote autonomous system number.
Step 5	address-family { <i>ipv4</i> <i>ipv6</i> } unicast Example: RP/0/RP0/CPU0: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:	Specifies the peers to whom the permanent network (path) is advertised.

	Command or Action	Purpose
	<pre>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# advertise permanent-network</pre>	
Step 7	commit	
Step 8	show bgp {ipv4 ipv6} unicast neighbor ip-address Example: <pre>RP/0/RP0/CPU0:router# show bgp ipv4 unicast neighbor 10.255.255.254</pre>	(Optional) Displays whether the neighbor is capable of receiving BGP permanent networks.

Enabling BGP Unequal Cost Recursive Load Balancing

Perform this task to enable unequal cost recursive load balancing for external BGP (eBGP), interior BGP (iBGP), and eiBGP and to enable BGP to carry link bandwidth attribute of the demilitarized zone (DMZ) link.

When the PE router includes the link bandwidth extended community in its updates to the remote PE through the Multiprotocol Interior BGP (MP-iBGP) session (either IPv4 or VPNv4), the remote PE automatically does load balancing if the **maximum-paths** command is enabled.

Unequal cost recursive load balancing happens across maximum eight paths only.



Note Enabling BGP unequal cost recursive load balancing feature is not supported on CPP based cards.

SUMMARY STEPS

1. **configure**
2. **router bgp as-number**
3. **address-family { ipv4 | ipv6 } unicast**
4. **maximum-paths { ebgp | ibgp | eibgp } maximum [unequal-cost]**
5. **exit**
6. **neighbor ip-address**
7. **dmz-link-bandwidth**
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp as-number Example: <pre>RP/0/RP0/CPU0: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.

	Command or Action	Purpose
Step 3	address-family { ipv4 ipv6 } unicast Example: RP/0/RP0/CPU0: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	maximum-paths { ebgp ibgp eibgp } maximum [unequal-cost] Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.
Step 6	neighbor ip-address Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router(config-bgp-nbr)# dmz-link-bandwidth	Originates a demilitarized-zone (DMZ) link-bandwidth extended community for the link to an eBGP/iBGP neighbor.
Step 8	commit	

Configuring VRF Dynamic Route Leaking

Perform these steps to import routes from default-VRF to non-default VRF or to import routes from non-default VRF to default VRF.

Before you begin

A route-policy is mandatory for configuring dynamic route leaking. Use the **route-policy route-policy-name** command in global configuration mode to configure a route-policy.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf_name*
3. **address-family** {*ipv4* | *ipv6*} **unicast**
4. Use one of these options:
 - **import from default-vrf route-policy** *route-policy-name* [**advertise-as-vpn**]
 - **export to default-vrf route-policy** *route-policy-name*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	vrf <i>vrf_name</i> Example: RP/0/RSP0/CPU0:PE51_CRS-9010(config)#vrf vrf_1	Enters VRF configuration mode.
Step 3	address-family { <i>ipv4</i> <i>ipv6</i> } unicast Example: RP/0/RP0/CPU0:router(config-vrf)#address-family ipv6 unicast	Enters VRF address-family configuration mode.
Step 4	Use one of these options: <ul style="list-style-type: none"> • import from default-vrf route-policy <i>route-policy-name</i> [advertise-as-vpn] • export to default-vrf route-policy <i>route-policy-name</i> Example: RP/0/RP0/CPU0:router(config-vrf-af)#import from default-vrf route-policy rpl_dynamic_route_import or RP/0/RP0/CPU0:router(config-vrf-af)#export to default-vrf route-policy rpl_dynamic_route_export	Imports routes from default-VRF to non-default VRF or from non-default VRF to default-VRF. <ul style="list-style-type: none"> • import from default-vrf—configures import from default-VRF to non-default-VRF. If the advertise-as-vpn option is configured, the paths imported from the default-VRF to the non-default-VRF are advertised to the PEs as well as to the CEs. If the advertise-as-vpn option is not configured, the paths imported from the default-VRF to the non-default-VRF are not advertised to the PE. However, the paths are still advertised to the CEs. • export to default-vrf—configures import from non-default-VRF to default VRF. The paths imported from the default-VRF are advertised to other PEs.
Step 5	commit	

What to do next

These **show bgp** command output displays information from the dynamic route leaking configuration:

- Use the **show bgp prefix** command to display the source-RD and the source-VRF for imported paths, including the cases when IPv4 or IPv6 unicast prefixes have imported paths.

- Use the **show bgp imported-routes** command to display IPv4 unicast and IPv6 unicast address-families under the default-VRF.

Configuring Resilient Per-CE Label Mode

Configuring Resilient Per-CE Label Mode Under VRF Address Family

Perform this task to configure resilient per-ce label mode under VRF address family.



Note Resilient per-CE 6PE label allocation is not supported on CRS-1 and CRS-3 routers, but supported only on ASR 9000 routers.

SUMMARY STEPS

1. **configure**
2. **router bgpas-number**
3. **vrfvrf-instance**
4. **address-family {ipv4 | ipv6} unicast**
5. **label mode per-ce**
6. Do one of the following:
 - **end**
 - **commit**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters global configuration mode.

Step 2 **router bgpas-number**

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 666  
RP/0/RP0/CPU0:router(config-bgp)#
```

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

Step 3 **vrfvrf-instance**

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# vrf vrf-pe
RP/0/RP0/CPU0:router(config-bgp-vrf)#
```

Configures a VRF instance.

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

Example:

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

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

Step 5 **label mode per-ce**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# label mode per-ce
RP/0/RP0/CPU0:router(config-bgp-vrf-af)#
```

Configures resilient per-ce label mode.

Step 6 Do one of the following:

- **end**
- **commit**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# end
```

or

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
 - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
 - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.
-

Configuring Resilient Per-CE Label Mode Using a Route-Policy

Perform this task to configure resilient per-ce label mode using a route-policy.



Note Resilient per-CE 6PE label allocation is not supported on CRS-1 and CRS-3 routers, but supported only on ASR 9000 routers.

SUMMARY STEPS

1. **configure**
2. **route-policy***policy-name*
3. **set label mode per-ce**
4. Do one of the following:
 - **end**
 - **commit**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters global configuration mode.

Step 2 **route-policy***policy-name*

Example:

```
RP/0/RP0/CPU0:router(config)# route-policy route1
RP/0/RP0/CPU0:router(config-rpl)#
```

Creates a route policy and enters route policy configuration mode.

Step 3 **set label mode per-ce**

Example:

```
RP/0/RP0/CPU0:router(config-rpl)# set label mode per-ce
RP/0/RP0/CPU0:router(config-rpl)#
```

Configures resilient per-ce label mode.

Step 4 Do one of the following:

- **end**
- **commit**

Example:


```
RP/0/RP0/CPU0:router(config-rpl)# end
```

or

```
RP/0/RP0/CPU0:router(config-rpl)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
 - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
 - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Implementing BGP

This section provides the following configuration examples:

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-falover 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 ipv4 multicast
    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
  address-family ipv6 multicast
    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
    address-family ipv4 multicast
  neighbor 10.0.101.61
    remote-as 65000
    address-family ipv4 unicast
    address-family ipv4 multicast
  neighbor 10.0.101.62
    remote-as 3
    address-family ipv4 unicast
      route-policy pass-all in
      route-policy pass-all out
    address-family ipv4 multicast
      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
    address-family ipv4 multicast
      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` command run in EXEC configuration mode:

```
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 131.108.0.0 and 192.31.7.0 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 131.108.234.2; 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 131.108.0.0 255.0.0.0
    network 192.31.7.0 255.0.0.0
    neighbor 131.108.200.1
      remote-as 167
    exit
  address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-out out
    neighbor 131.108.234.2
      remote-as 109
    exit
  address-family ipv4 unicast
    neighbor 150.136.64.19
      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 171.69.232.55 and 171.69.232.56 get the local preference, next hop, and MED unmodified in the updates. The router at 160.69.69.1 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 171.69.232.55
    remote-as 6002
  exit
  address-family ipv4 unicast
    neighbor 171.69.232.56
    remote-as 6003
  exit
  address-family ipv4 unicast
    neighbor 160.69.69.1
    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 170.70.70.1 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 170.70.70.1
    remote-as 6002
  exit
  address-family ipv4 unicast
    neighbor 171.69.232.57
    remote-as 6001
  exit
  address-family ipv4 unicast
    neighbor 171.69.232.56
    remote-as 6003
  exit
  address-family ipv4 unicast
```

```
neighbor 199.69.99.2
  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 200.200.200.200 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 171.69.232.57
  remote-as 6001
  exit
address-family ipv4 unicast
  neighbor 171.69.232.55
  remote-as 6002
  exit
address-family ipv4 unicast
  neighbor 200.200.200.200
  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 200.200.200.205 from autonomous system 701 in the same example. Neighbor 171.69.232.56 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 171.69.232.56
  remote-as 666
  exit
address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
  exit
address-family ipv4 unicast
  neighbor 200.200.200.205
  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 for both unicast and multicast prefixes:

```

router bgp 140
  address-family ipv4 unicast
  neighbor 10.1.1.1
  remote-as 140
  address-family ipv4 unicast
  route-reflector-client
  exit
  address-family ipv4 multicast
  route-reflector-client

```

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 vpv4 unicast
  address-family ipv4 mdt

  !
  neighbor 1.1.1.1

  remote-as 11
  update-source Loopback0
  address-family ipv4 unicast
  address-family vpv4 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 120
  address-family ipv4 unicast
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy bgp_add_path
  !
  !
end
```

Allocated Local Label Retention: Example

The following example shows how to retain the previously allocated local label for the primary path on the primary PE for 10 minutes after reconvergence:

```
router bgp 100
  address-family l2vpn vpls-vpws
  retain local-label 10
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
```


Topology Scenario

This figure shows a basic scenario for per neighbor TCP MSS configuration.



R1 Configuration:

```

router bgp 1
  bgp router-id 10.0.0.1
  address-family ipv4 unicast
  !
  neighbor-group n1
    tcp mss 100
    address-family ipv4 unicast
  !
  !
  neighbor 10.0.0.2
    remote-as 1
    use neighbor-group n1
    address-family ipv4 unicast
  !
  !
  !
  
```

R2 Configuration:

```

router bgp 1
  bgp router-id 10.0.0.2
  address-family ipv4 unicast
  !
  neighbor 10.0.0.1
    remote-as 1
    address-family ipv4 unicast
  !
  !
  !
  
```

Configure Per Neighbor TCP MSS: Example

The following example shows how to configure per neighbor TCP MSS under neighbor group:

```

router bgp 1
  bgp router-id 10.0.0.1
  address-family ipv4 unicast
  !
  neighbor-group n1
    tcp mss 500
    address-family ipv4 unicast
  !
  !
  neighbor 10.0.0.2
    remote-as 1
    use neighbor-group n1
    address-family ipv4 unicast
  !
  
```

```
!
!
!
end
```

Disable Per Neighbor TCP MSS: Example

The following example shows how to configure TCP MSS under neighbor group and configure inheritance disable under one of the neighbors inheriting the TCP MSS value:

```
router bgp 1
  bgp router-id 10.0.0.1
  address-family ipv4 unicast
  !
  neighbor-group n1
  tcp mss 500
  address-family ipv4 unicast
  !
  !
  neighbor 10.0.0.2
  remote-as 1
  use neighbor-group n1
  tcp mss inheritance-disable
  address-family ipv4 unicast
  !
  !
  !
end
```

Unconfigure TCP MSS: Example

The following example shows how to unconfigure TCP MSS:

```
RP/0/0/CPU0:ios(config)#router bgp 1
RP/0/0/CPU0:ios(config-bgp)#neighbor-group n1
RP/0/0/CPU0:ios(config-bgp-nbrgrp)#no tcp mss 500
RP/0/0/CPU0:ios(config-bgp-nbrgrp)#commit
```

Verify Per Neighbor TCP MSS: Examples

The following example shows how to verify the per neighbor TCP MSS feature on a router:

```
RP/0/0/CPU0:ios#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 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

```

VRF Dynamic Route Leaking Configuration: Example

These examples show how to configure VRF dynamic route leaking:

Import Routes from default-VRF to non-default-VRF

```

vrf vrf_1
  address-family ipv6 unicast
    import from default-vrf route-policy rpl_dynamic_route_import
  !
end

```

Import Routes from non-default-VRF to default-VRF

```

vrf vrf_1
  address-family ipv6 unicast
    export to default-vrf route-policy rpl_dynamic_route_export
  !
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.

Where to Go Next

For detailed information about BGP commands, see *Routing Command Reference for Cisco CRS Routers*

Additional References

The following sections provide references related to implementing BGP.

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Routing Command Reference for Cisco CRS Routers</i>
Cisco Express Forwarding (CEF) commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>IP Addresses and Services Command Reference for Cisco CRS Routers</i>
MPLS VPN configuration information.	<i>MPLS Configuration Guide for the Cisco CRS Routers</i>
Bidirectional Forwarding Detection (BFD)	<i>Interface and Hardware Component Configuration Guide for Cisco CRS Routers</i> and <i>Interface and Hardware Component Command Reference for Cisco CRS Routers</i>
Task ID information.	Configuring AAA Services on Cisco IOS XR Software module of <i>System Security Configuration Guide for Cisco CRS Routers</i>

Standards

Standards	Title
draft-bonica-tcp-auth-05.txt	<i>Authentication for TCP-based Routing and Management Protocols</i> , by R. Bonica, B. Weis, S. Viswanathan, A. Lange, O. Wheeler
draft-ietf-idr-bgp4-26.txt	<i>A Border Gateway Protocol 4</i> , by Y. Rekhter, T.Li, S. Hares
draft-ietf-idr-bgp4-mib-15.txt	<i>Definitions of Managed Objects for the Fourth Version of Border Gateway Protocol (BGP-4)</i> , by J. Hass and S. Hares
draft-ietf-idr-cease-subcode-05.txt	<i>Subcodes for BGP Cease Notification Message</i> , by Enke Chen, V. Gillet
draft-ietf-idr-avoid-transition-00.txt	<i>Avoid BGP Best Path Transitions from One External to Another</i> , by Enke Chen, Srihari Sangli
draft-ietf-idr-as4bytes-12.txt	<i>BGP Support for Four-octet AS Number Space</i> , by Quazar Vohra, Enke Chen
draft-nalawade-idr-mdt-safi-03.txt	<i>MDT SAFI</i> , by Gargi Nalawade and Arjun Sreekantiah

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

RFCs

RFCs	Title
RFC 1700	Assigned Numbers
RFC 1997	BGP Communities Attribute
RFC 2385	Protection of BGP Sessions via the TCP MD5 Signature Option
RFC 2439	BGP Route Flap Damping
RFC 2545	Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing
RFC 2796	BGP Route Reflection - An Alternative to Full Mesh IBGP

RFCs	Title
RFC 2858	Multiprotocol Extensions for BGP-4
RFC 2918	Route Refresh Capability for BGP-4
RFC 3065	Autonomous System Confederations for BGP
RFC 3392	Capabilities Advertisement with BGP-4
RFC 4271	A Border Gateway Protocol 4 (BGP-4)
RFC 4364	BGP/MPLS IP Virtual Private Networks (VPNs)
RFC 4724	<i>Graceful Restart Mechanism for BGP</i>
RFC 2784	Generic Routing Encapsulation (GRE)

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport



CHAPTER 3

Implementing BGP Flowspec

Flowspec specifies procedures for the distribution of flow specification rules via BGP and defines procedure to encode flow specification rules as Border Gateway Protocol Network Layer Reachability Information (BGP NLRI) which can be used in any application. It also defines application for the purpose of packet filtering in order to mitigate (distributed) denial of service attacks.



Note For more information about BGP Flowspec and complete descriptions of the BGP Flowspec commands listed in this module, see the *BGP Flowspec Commands* chapter in the *Routing Command Reference for Cisco CRS Routers*.

Feature History for Implementing BGP Flowspec

Release 5.2.0	This feature was introduced.
------------------	------------------------------

- [BGP Flow Specification, on page 215](#)

BGP Flow Specification

The BGP flow specification (flowspec) feature allows you to rapidly deploy and propagate filtering and policing functionality among a large number of BGP peer routers to mitigate the effects of a distributed denial-of-service (DDoS) attack over your network.

In traditional methods for DDoS mitigation, such as RTBH (remotely triggered blackhole), a BGP route is injected advertising the website address under attack with a special community. This special community on the border routers sets the next hop to a special next hop to discard/null, thus preventing traffic from suspect sources into your network. While this offers good protection, it makes the Server completely unreachable.

BGP flowspec, on the other hand, allows for a more granular approach and lets you effectively construct instructions to match a particular flow with source, destination, L4 parameters and packet specifics such as length, fragment and so on. Flowspec allows for a dynamic installation of an action at the border routers to either:

- Drop the traffic
- Inject it in a different VRF for analysis or

- Allow it, but police it at a specific defined rate

Thus, instead of sending a route with a special community that the border routers must associate with a next hop to drop in their route policy language, BGP flowspec sends a specific flow format to the border routers instructing them to create a sort of ACL with class-map and policy-map to implement the rule you want advertised. In order to accomplish this, BGP flowspec adds a new NLRI (network layer reachability information) to the BGP protocol. [Information About Implementing BGP Flowspec](#), on page 217 provides details on flow specifications, supported matching criteria and traffic filtering action.

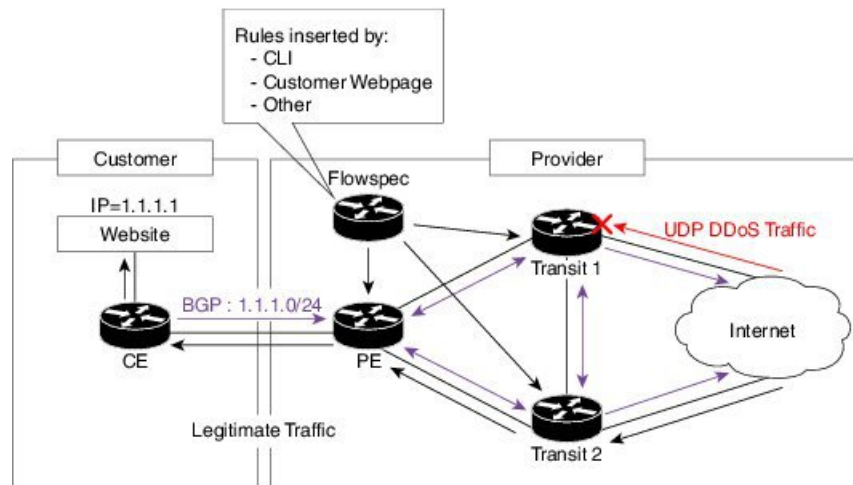
Limitations

These limitations apply for BGP flowspec:

- Flowspec is not supported on Cisco CRS-1 line cards (MSC-A, MSC-B, FP-40).
- Flowspec is not supported on subscriber and satellite interfaces.
- A maximum of five multi-value range can be specified in a flowspec rule.
- A mix of address families is not allowed in flowspec rules.
- In multiple match scenario, only the first matching flowspec rule will be applied.
- There are 1500 TCAM entries. If each rule translates into one entry, then a maximum of 1500 flowspec rules are supported per system.
- QoS takes precedence over BGP flowspec.
- You cannot do explicit matching with respect to MPLS.
- This feature does not function if another configured feature causes all the IPv6 parsing to go to slowpath. For example, this feature does not function when port mirroring is configured.
- The feature is supported on Cisco ASR 9000 Third Generation Ethernet Line Cards. Other line cards in the router do not support this feature.
- If there is a bundle in the router, and if one of the line card interfaces in the bundle belongs to a line card that is not Cisco ASR 9000 Third Generation Ethernet Line Card, you may observe inconsistent behaviour.
- You cannot enable this feature on satellite interfaces.
- This feature supports TCP, UDP, and ICMPv6.

BGP Flowspec Conceptual Architecture

In this illustration, a Flowspec router (controller) is configured on the Provider Edge with flows (match criteria and actions). The Flowspec router advertises these flows to the other edge routers and the AS (that is, Transit 1, Transit 2 and PE). These transit routers then install the flows into the hardware. Once the flow is installed into the hardware, the transit routers are able to do a lookup to see if incoming traffic matches the defined flows and take suitable action. The action in this scenario is to 'drop' the DDoS traffic at the edge of the network itself and deliver only clean and legitimate traffic to the Customer Edge.



The ensuing section provides an example of the CLI configuration of how flowspec works. First, on the Flowspec router you define the match-action criteria to take on the incoming traffic. This comprises the PBR portion of the configuration. The **service-policy type** defines the actual PBR policy and contains the combination of match and action criteria which must be added to the flowspec. In this example, the policy is added under address-family IPv4, and hence it is propagated as an IPv4 flowspec rule.

Flowspec router CLI example:

```
class-map type traffic match-all cml
  match source-address ipv4 100.0.0.0/24

policy-map type pbr pm1
  class type traffic cml
  drop

flowspec
  address-family ipv4
  service-policy type pbr pm0
```

Transient router CLI:

```
flowspec
  address-family ipv4
  service-policy type pbr pm1
```

For detailed procedural information and commands used for configuring Flowspec, see [Configuring BGP Flowspec with ePBR, on page 224](#).

Information About Implementing BGP Flowspec

To implement BGP Flowspec, you need to understand the following concepts:

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 is said to match the defined flow if it matches all the specified criteria. A given flow may be associated with a set of attributes, depending on the particular application; such attributes may or may not include reachability information (that is, NEXT_HOP).

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. [Supported Matching Criteria and Actions, on page 218](#) and [Traffic Filtering Actions, on page 221](#) provides information on the supported match (tuple definitions) and action parameters.

Supported Matching Criteria and Actions

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 on receipt. 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 (AND) of all the components present in the specification. The following are the supported component types or tuples that you can define:

Tuple definition possibilities

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} address/mask length</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} address/mask length</p>	Prefix length

Type 3	IPv4 last next header or IPv6Protocol	<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>Type 3: match protocol {<i>protocol-value</i> <i>min-value</i> -<i>max-value</i>}</p>	Multi value range
Type 4	IPv4 or IPv6 source or destination port	<p>Defines a list of {operation, value} pairs that matches source or destination TCP/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>Encoding: <type (1 octet), [op, value]+></p> <p>Syntax:</p> <p>match {ipv4 ipv6}icmp-code {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>	Bit mask
Type 10	IPv4 or IPv6 Packet length	<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, where the two most significant bits are zero and the six least significant bits contain the DSCP value.</p> <p>Encoding: <type (1 octet), [op, value]+></p> <p>Syntax:</p> <p>match dscp {<i>dscp-value</i> <i>min-value</i> -<i>max-value</i>}</p>	Multi value range
Type 12	IPv4 or IPv6 Fragmentation bits	<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 [<i>is-fragment</i>]</p>	Bit mask

In a given flowspec rule, multiple action combinations can be specified without restrictions. However, address family mixing between matching criterion and actions are not allowed. For example, IPv4 matches cannot be combined with IPv6 actions and vice versa.



Note Redirect IP Nexthop is only supported in default VRF cases.

[Traffic Filtering Actions, on page 221](#) provides information on the actions that can be associated with a flow. [Configuring BGP Flowspec with ePBR, on page 224](#) explains the procedure to configure BGP flowspec with the required tuple definitions and action sequences.

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:

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 purely informational. The remaining 4 octets carry the rate information in IEEE floating point [IEEE.754.1985] format, units being 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>

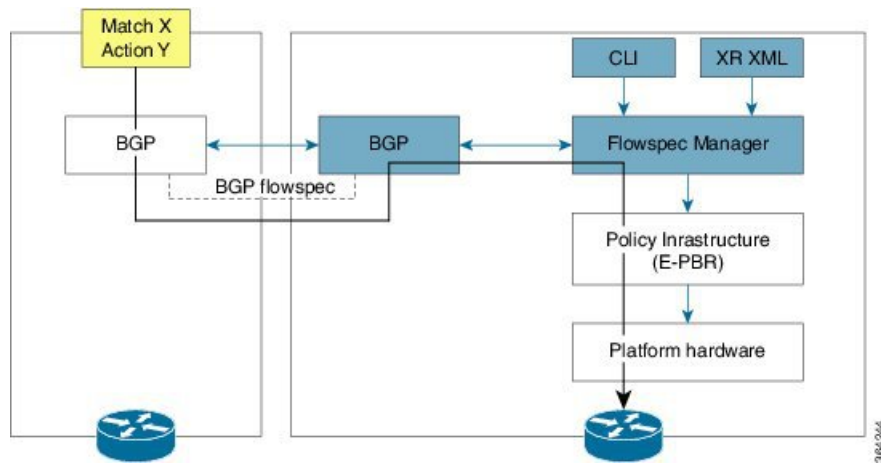
0x8008	redirect-vrf	Redirect VRF	<p>The redirect extended community allows the traffic to be redirected to a VRF routing instance that lists the specified route-target in its import policy. If several local instances match this criteria, the choice between them is a local matter (for example, the instance with the lowest Route Distinguisher value can be elected). This extended community uses the same encoding as the Route Target extended community [RFC4360].</p> <p>Command syntax based on route-target</p> <pre>redirect {ipv6} extcommunity rt <route_target_string></pre>
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> <pre>set dscp <6 bit value> set ipv6 traffic-class <8 bit value></pre>
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 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>Command syntax</p> <pre>redirect {ipv6} next-hop <ipv4/v6 address> {ipv4/v6 address}</pre>

[Configure a Class Map, on page 225](#) explains how you can configure specific match criteria for a class map.

BGP Flowspec Client-Server (Controller) Model and Configuration with ePBR

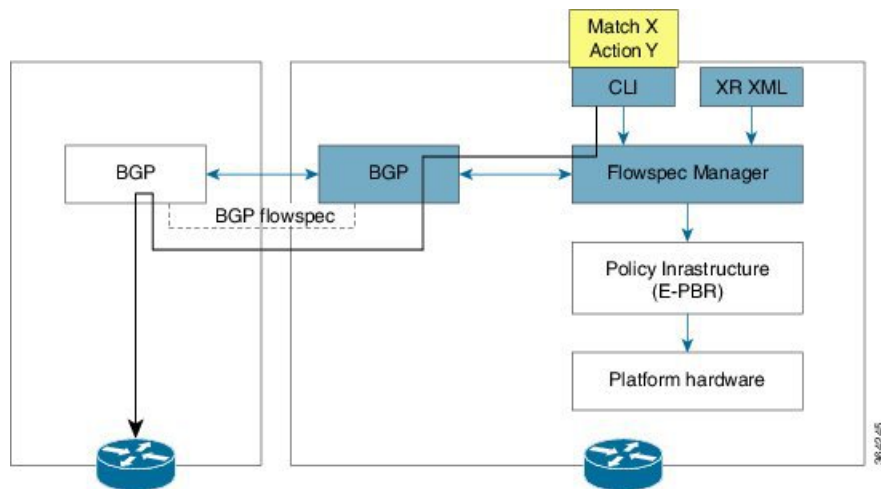
The BGP Flowspec model comprises of a Client and a Server (Controller). The Controller is responsible for sending or injecting the flowspec NRI entry. The client (acting as a BGP speaker) receives that NRI and programs the hardware forwarding to act on the instruction from the Controller. An illustration 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 that entry for NRLI injection.

BGP Flowspec Configuration

- **BGP-side:** You must enable the new address family for advertisement. This procedure is applicable for both the Client and the Controller. [Enable BGP Flowspec, on page 224](#) explains the procedure.
- **Client-side:** No specific configuration, except availability of a flowspec-enabled peer.
- **Controller-side:** This includes the policy-map definition and the association to the ePBR configuration consists of two procedures: the class definition, and using that class in ePBR to define the action. The following topics explain the procedure:
 - [Configure a Policy Map, on page 227](#)
 - [Configure a Class Map, on page 225](#)
 - [Link BGP Flowspec to ePBR Policies, on page 229](#)

Configuring BGP Flowspec with ePBR

The following sections explain the procedures for configuring BGP flowspec with ePBR.

Use the following procedures to enable and configure the BGP flowspec feature:

- [Enable BGP Flowspec, on page 224](#)
- [Configure a Class Map, on page 225](#)
- [Configure a Policy Map, on page 227](#)
- [Link BGP Flowspec to ePBR Policies , on page 229](#)



Note To save configuration changes, you must commit changes when the system prompts you.

Enable BGP Flowspec

You must enable the address family for propagating the BGP flowspec policy on both the Client and Server using the following steps:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** | **vpn4** | **vpn6** } **flowspec**
4. **exit**
5. **neighbor** *ip-address*
6. **remote-as** *as-number*
7. **address-family** { **ipv4** | **ipv6** } **flowspec**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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 vpn4 vpn6 } flowspec Example: RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 flowspec	Specifies either the IPv4, IPv6, vpn4 or vpn6 address family and enters address family configuration submenu, and initializes the global address family for flowspec policy mapping.
Step 4	exit Example:	Returns the router to BGP configuration mode.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-bgp-af)# exit	
Step 5	neighbor <i>ip-address</i> Example: RP/0/RP0/CPU0:router(config-bgp)#neighbor 1.1.1.1	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 6	remote-as <i>as-number</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)#remote-as 100	Assigns a remote autonomous system number to the neighbor.
Step 7	address-family { <i>ipv4</i> <i>ipv6</i> } flowspec Example: RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 flowspec	Specifies an address family and enters address family configuration submenu, and initializes the global address family for flowspec policy mapping.

Configuring an address family for flowspec policy mapping: Example

```

router bgp 100

  address-family ipv4 flowspec
  ! Initializes the global address family
  address-family ipv6 flowspec
  !
  neighbor 1.1.1.1
    remote-as 100
  address-family ipv4 flowspec
  ! Ties it to a neighbor configuration
  address-family ipv6 flowspec
  !

```

Configure a Class Map

In order to associate the ePBR configuration to BGP flowspec you must perform these sub-steps: define the class and use that class in ePBR to define the action. The steps to define the class include:

SUMMARY STEPS

1. **configure**
2. **class-map [type traffic] [match-all] *class-map-name***
3. **match *match-statement***

4. end-class-map

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	class-map [type traffic] [match-all] <i>class-map-name</i> Example: <pre>RP/0/RP0/CPU0:router(config)# class-map type traffic match all class1</pre>	Creates a class map to be used for matching packets to the class whose name you specify and enters the class map configuration mode. If you specify match-any , one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. This is the default. If you specify match-all , the traffic must match all the match criteria.
Step 3	match <i>match-statement</i> Example: <pre>RP/0/RP0/CPU0:router(config-cmap)# match protocol ipv4 1 60</pre>	<p>Configures the match criteria for a class map on the basis of the statement specified. Any combination of tuples 1-13 match statements can be specified here. The tuple definition possibilities include:</p> <ul style="list-style-type: none"> • Type 1: match destination-address {<i>ipv4 ipv6</i>} <i>address/mask length</i> • Type 2: match source-address {<i>ipv4 ipv6</i>} <i>address/mask length</i> • Type 3: match protocol {<i>protocol-value min-value -max-value</i>} <p>Note In case of IPv6, it will map to last next-header.</p> <ul style="list-style-type: none"> • Type 4: Create two class-maps: one with source-port and another with destination-port: <ul style="list-style-type: none"> • match source-port {<i>source-port-value min-value -max-value</i>} • match destination-port {<i>destination-port-value min-value -max-value</i>} <p>Note These are applicable only for TCP and UDP protocols.</p> • Type 5: match destination-port {<i>destination-port-value [min-value - max-value]</i>} • Type 6: match source-port {<i>source-port-value [min-value - max-value]</i>} • Type 7: match {<i>ipv4 ipv6</i>} icmp-code {<i>value min-value -max-value</i>} • Type 8: match {<i>ipv4 ipv6</i>} icmp-type {<i>value min-value -max-value</i>}

	Command or Action	Purpose
		<ul style="list-style-type: none"> • Type 9: match tcp-flag <i>value bit-mask mask_value</i> • Type 10: matchpacket length {<i>packet-length-value min-value -max-value</i>} • Type 11: match dscp {<i>dscp-value min-value -max-value</i>} • Type 12: match fragment-type {dont-fragment is-fragment first-fragment last-fragment} • Type 13: match ipv6 flow-label ipv4 flow-label {<i>value min-value -max-value</i>} <p>Supported Matching Criteria and Actions, on page 218 provides conceptual information on these match parameters.</p> <p><i>BGP Flowspec Commands</i> in the <i>Routing Command Reference for Cisco CRS Routers</i> guide provides additional details on the various commands used for BGP flowspec configuration.</p>
Step 4	end-class-map Example: <pre>RP/0/RP0/CPU0:router(config-cmap)# end-class-map</pre>	Ends the class map configuration and returns the router to global configuration mode.

What to do next

Associate the class defined in this procedure to a PBR policy as described in [Configure a Policy Map, on page 227](#).

Configure a Policy Map

This procedure helps you define a policy map and associate it with traffic class you configured previously in [Configure a Class Map, on page 225](#).

SUMMARY STEPS

1. **configure**
2. **policy-map type pbr** *policy-map*
3. **class** *class-name*
4. **class type traffic** *class-name*
5. *action*
6. **exit**
7. **end-policy-map**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	

	Command or Action	Purpose
Step 2	<p>policy-map type pbr <i>policy-map</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config)# policy-map type pbr policycp1</pre>	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	<p>class <i>class-name</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-pmap)# class class1</pre>	Specifies the name of the class whose policy you want to create or change.
Step 4	<p>class type traffic <i>class-name</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-pmap)# class type traffic classcl</pre>	Associates a previously configured traffic class with the policy map, and enters control policy-map traffic class configuration mode.
Step 5	<p><i>action</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-pmap-c)# set dscp 5</pre>	<p>Define extended community actions as per your requirement. The options include:</p> <ul style="list-style-type: none"> • Traffic rate: police rate <i>rate</i> • Redirect VRF: redirect { ipv4ipv6 } extcommunity rt <i>route_target_string</i> • Traffic Marking: set { dscp rate destination-address {ipv4 ipv6} <i>8-bit value</i>} • Redirect IP NH: redirect { ipv4ipv6 } nexthop <i>ipv4 addressipv6 address</i> { <i>ipv4 addressipv6 address</i>} <p>Traffic Filtering Actions, on page 221 provides conceptual information on these extended community actions.</p>
Step 6	<p>exit</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-pmap-c)# exit</pre>	Returns the router to policy map configuration mode.
Step 7	<p>end-policy-map</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-cmap)# end-policy-map</pre>	Ends the policy map configuration and returns the router to global configuration mode.

What to do next

Perform VRF and flowspec policy mapping for distribution of flowspec rules using the procedure explained in [Link BGP Flowspec to ePBR Policies](#), on page 229

Link BGP Flowspec to ePBR Policies

For BGP flowspec, an ePBR policy is applied on a per VRF basis, and this policy is applied on all the interfaces that are part of the VRF. If you have already configured a ePBR policy on an interface, it will not be overwritten by the BGP flowspec policy. If you remove the policy from an interface, ePBR infrastructure will automatically apply BGP flowspec policy on it, if one was active at the VRF level.



Note At a time only one ePBR policy can be active on an interface.

SUMMARY STEPS

1. **configure**
2. **flowspec**
3. **local-install interface-all**
4. **address-family ipv4**
5. **local-install interface-all**
6. **service-policy type pbr *policy-name***
7. **exit**
8. **address-family ipv6**
9. **local-install interface-all**
10. **service-policy type pbr *policy-name***
11. **vrf *vrf-name***
12. **address-family ipv4**
13. **local-install interface-all**
14. **service-policy type pbr *policy-name***
15. **exit**
16. **address-family ipv6**
17. **local-install interface-all**
18. **service-policy type pbr *policy-name***
19. **commit**
20. **exit**
21. **show flowspec { *afi-all* | *client* | *ipv4* | *ipv6* | *summary* | *vrf***

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	flowspec Example:	Enters the flowspec configuration mode.

	Command or Action	Purpose
	<pre>RP/0/RP0/CPU0:router(config)# flowspec</pre>	
Step 3	<p>local-install interface-all</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-flowspec)# local-install interface-all</pre>	(Optional) Installs the flowspec policy on all interfaces.
Step 4	<p>address-family ipv4</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-flowspec)# address-family ipv4</pre>	Specifies either an IPv4 address family and enters address family configuration submenu.
Step 5	<p>local-install interface-all</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-flowspec-af)# local-install interface-all</pre>	(Optional) Installs the flowspec policy on all interfaces under the subaddress family.
Step 6	<p>service-policy type pbr <i>policy-name</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-flowspec-af)# service-policy type pbr policys1</pre>	Attaches a policy map to an IPv4 interface to be used as the service policy for that interface.
Step 7	<p>exit</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-flowspec-af)# exit</pre>	Returns the router to flowspec configuration mode.
Step 8	<p>address-family ipv6</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-flowspec)# address-family ipv6</pre>	Specifies an IPv6 address family and enters address family configuration submenu.
Step 9	<p>local-install interface-all</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-flowspec-af)# local-install interface-all</pre>	(Optional) Installs the flowspec policy on all interfaces under the subaddress family.

	Command or Action	Purpose
Step 10	service-policy type pbr <i>policy-name</i> Example: <pre>RP/0/RP0/CPU0:router(config-flowspec-af)# service-policy type pbr policys1</pre>	Attaches a policy map to an IPv6 interface to be used as the service policy for that interface.
Step 11	vrf <i>vrf-name</i> Example: <pre>RP/0/RP0/CPU0:router(config-flowspec)# vrf vrf1</pre>	Configures a VRF instance and enters VRF flowspec configuration submode.
Step 12	address-family ipv4 Example: <pre>RP/0/RP0/CPU0:router(config-flowspec-vrf)# address-family ipv4</pre>	Specifies an IPv4 address family and enters address family configuration submode.
Step 13	local-install interface-all Example: <pre>RP/0/RP0/CPU0:router(config-flowspec-vrf-af)# local-install interface-all</pre>	(Optional) Installs the flowspec policy on all interfaces under the subaddress family.
Step 14	service-policy type pbr <i>policy-name</i> Example: <pre>RP/0/RP0/CPU0:router(config-flowspec-vrf-af)# service-policy type pbr policys1</pre>	Attaches a policy map to an IPv4 interface to be used as the service policy for that interface.
Step 15	exit Example: <pre>RP/0/RP0/CPU0:router(config-flowspec-vrf-af)# exit</pre>	Returns the router to VRF flowspec configuration submode.
Step 16	address-family ipv6 Example: <pre>RP/0/RP0/CPU0:router(config-flowspec-vrf)# address-family ipv6</pre>	Specifies either an IPv6 address family and enters address family configuration submode.
Step 17	local-install interface-all Example:	(Optional) Installs the flowspec policy on all interfaces under the subaddress family.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-flowspec-vrf-af)# local-install interface-all	
Step 18	service-policy type pbr <i>policy-name</i> Example: RP/0/RP0/CPU0:router(config-flowspec-vrf-af)# service-policy type pbr policys1	Attaches a policy map to an IPv6 interface to be used as the service policy for that interface.
Step 19	commit	
Step 20	exit Example: RP/0/RP0/CPU0:router(config-flowspec-vrf-af)# exit	Returns the router to flowspec configuration mode.
Step 21	show flowspec { afi-all client ipv4 ipv6 summary vrf } Example: RP/0/RP0/CPU0:router# show flowspec vrf vrf1 ipv4 summary	(Optional) Displays flowspec policy applied on an interface.

Verify BGP Flowspec

Use these different **show** commands to verify your flowspec configuration. For instance, you can use the associated flowspec and BGP show commands to check whether flowspec rules are present in your table, how many rules are present, the action that has been taken on the traffic based on the flow specifications you have defined and so on.

SUMMARY STEPS

1. **show processes flowspec_mgr location all**
2. **show flowspec summary**
3. **show flowspec vrf** *vrf_name* | **all** { **afi-all** | **ipv4** | **ipv6** }
4. **show bgp ipv4 flowspec**

DETAILED STEPS

	Command or Action	Purpose
Step 1	show processes flowspec_mgr location all Example:	Specifies whether the flowspec process is running on your system or not. The flowspec manager is responsible for

	Command or Action	Purpose
	<pre># show processes flowspec_mgr location all node: node0_3_CPU0 Job Id: 10 PID: 43643169 Executable path: /disk0/iosxr-fwding-5.2.CSC33695-015.i/bin/flowspec_mgr Instance #: 1 Version ID: 00.00.0000 Respawn: ON Respawn count: 331 Max. spawns per minute: 12 Last started: Wed Apr 9 10:42:13 2014 Started on config: cfg/gl/flowspec/ Process group: central-services core: MAINMEM startup_path: /pkg/startup/flowspec_mgr.startup Ready: 1.113s Process cpu time: 0.225 user, 0.023 kernel, 0.248 total JID TID CPU Stack pri state TimeInState HR:MM:SS:MSEC NAME 1082 1 0 112K 10 Receive 2:50:23:0508 0:00:00:0241 flowspec_mgr 1082 2 1 112K 10 Sigwaitinfo 2:52:42:0583 0:00:00:0000 flowspec_mgr</pre>	<p>creating, distributing and installing the flowspec rules on the hardware.</p>
<p>Step 2</p>	<p>show flowspec summary</p> <p>Example:</p> <pre># show flowspec summary FlowSpec Manager Summary: Tables: 2 Flows: 1 RP/0/3/CPU0:RA01_R4#</pre>	<p>Provides a summary of the flowspec rules present on the entire node. In this example, the 2 table indicate that IPv4 and IPv6 has been enabled, and a single flow has been defined across the entire table.</p>
<p>Step 3</p>	<p>show flowspec vrf <i>vrf_name</i> all { afi-all ipv4 ipv6 }</p> <p>Example:</p> <pre># show flowspec vrf default ipv4 summary Flowspec VRF+AFI table summary: VRF: default AFI: IPv4 Total Flows: 1 Total Service Policies: 1 RP/0/3/CPU0:RA01_R4# ----- # show flowspec vrf default ipv6 summary Flowspec VRF+AFI table summary: VRF: default AFI: IPv6 Total Flows: 0 Total Service Policies: 0 RP/0/3/CPU0:RA01_R4# ----- # show flowspec vrf all afi-all summary</pre>	<p>In order to obtain more granular information on the flowspec, you can filter the show commands based on a particular address-family or by a specific VRF name. In this example, 'vrf default' indicates that the flowspec has been defined on the default table. The 'IPv4 summary' shows the IPv4 flowspec rules present on that default table. As there are no IPv6s configured, the value shows 'zero' for ipv6 summary 'Table Flows' and 'Policies' parameters. 'VRF all' displays information across all the VRFs configured on the table and afi-all displays information for all address families (IPv4 and IPv6).</p> <p>The detail option displays the 'Matched', 'Transmitted,' and 'Dropped' fields. These can be used to see if the flowspec rule you have defined is in action or not. If there is any traffic that takes this match condition, it indicates if any action has been taken (that is, how many packets were matched and whether these packets have been transmitted or dropped).</p>

	Command or Action	Purpose
	<pre>Flowspec VRF+AFI table summary: VRF: default AFI: IPv4 Total Flows: 1 Total Service Policies: 1 VRF: default AFI: IPv6 Total Flows: 0 Total Service Policies: 0 ----- # show flowspec vrf default ipv4 Dest:110.1.1.0/24, Source:10.1.1.0/24,DPort:>=120<=130, SPort:>=25<=30,DSCP:=30 detail AFI: IPv4 Flow :Dest:110.1.1.0/24,Source:10.1.1.0/24, DPort:>=120<=130,SPort:>=25<=30,DSCP:=30 Actions :Traffic-rate: 0 bps (bgp.1) Statistics (packets/bytes) Matched : 0/0 Transmitted : 0/0 Dropped : 0/0</pre>	
Step 4	<pre>show bgp ipv4 flowspec Example: # show bgp ipv4 flowspec Dest:110.1.1.0/24,Source:10.1.1.0/24, DPort:>=120<=130,SPort:>=25<=30,DSCP:=30/208 BGP routing table entry for Dest:110.1.1.0/24, Source:10.1.1.0/24,Proto:=47,DPort:>=120<=130,SPort:>=25<=30,DSCP:=30/208 <snip> Paths: (1 available, best #1) Advertised to update-groups (with more than one peer): 0.3 Path #1: Received by speaker 0 Advertised to update-groups (with more than one peer): 0.3 Local 0.0.0.0 from 0.0.0.0 (3.3.3.3) Origin IGP, localpref 100, valid, redistributed, best, group-best Received Path ID 0, Local Path ID 1, version 42 Extended community: FLOWSPEC Traffic-rate:100,0</pre>	<p>Use this command to verify if a flowspec rule configured on the controller router is available on the BGP side. In this example, 'redistributed' indicates that the flowspec rule is not internally originated, but one that has been redistributed from the flowspec process to BGP. The extended community (BGP attribute used to send the match and action criteria to the peer routers) you have configured is also displayed here. In this example, the action defined is to rate limit the traffic.</p>

Preserving Redirect Nexthop

You can explicitly configure redirect nexthop as part of the route specification. Redirect nexthop is encoded as the MP_REACH nexthop in the BGP flowspec NLRI along with the associated extended community. Recipient of such a flowspec route redirects traffic as per FIB lookup for the redirect nexthop, the nexthop can possibly resolve over IP or MPLS tunnel. As the MP_REACH nexthop can be overwritten at a eBGP

boundary, for cases where the nexthop connectivity spans multiple AS's, the nexthop can be preserved through the use of the unchanged knob.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **address-family** { **ipv4** | **ipv6** }
5. **flowspec next-hop unchanged**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: <pre>RP/0/RP0/CPU0:router(config)# router bgp 100</pre>	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: <pre>RP/0/RP0/CPU0:router(config)# router bgp 100 neighbor 1.1.1.1</pre>	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 } Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# router bgp 100 neighbor 1.1.1.1 address-family ipv4</pre>	Specifies either the IPv4 or IPv6 address family and enters address family configuration submenu, and initializes the global address family.
Step 5	flowspec next-hop unchanged Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# router bgp 100 neighbor 1.1.1.1 address-family ipv4 flowspec next-hop unchanged</pre>	Preserves the next-hop for the flowspec unchanged.

Validate BGP Flowspec

BGP Flowspec validation is enabled by default for flowspec SAFI routes for IPv4 or IPv6. VPN routes are not subject to the flow validation. A flow specification NLRI is validated to ensure that any one of the following conditions holds true for the functionality to work:

- The originator of the flow specification matches the originator of the best-match unicast route for the destination prefix embedded in the flow specification.

- There are no more specific unicast routes, when compared with the flow destination prefix, that have been received from a different neighboring AS than the best-match unicast route, which has been determined in the previous condition.
- The AS_PATH and AS4_PATH attribute of the flow specification are empty.
- The AS_PATH and AS4_PATH attribute of the flow specification does not contain AS_SET and AS_SEQUENCE segments.

Any path which does not meet these conditions, is appropriately marked by BGP and not installed in flowspec manager. Additionally, BGP enforces that the last AS added within the AS_PATH and AS4_PATH attribute of a EBGP learned flow specification NLRI must match the last AS added within the AS_PATH and AS4_PATH attribute of the best-match unicast route for the destination prefix embedded in the flow specification. Also, when the redirect-to-IP extended community is present, by default, BGP enforces the following check when receiving a flow-spec route from an eBGP peer:

If the flow-spec route has an IP next-hop X and includes a redirect-to-IP extended community, then the BGP speaker discards the redirect-to-ip extended community (and not propagate it further with the flow-spec route) if the last AS in the AS_PATH or AS4_PATH attribute of the longest prefix match for X does not match the AS of the eBGP peer.

[Disable Flowspec Redirect and Validation, on page 237](#) explains the procedure to disable BGP flowspec validation.

Disabling BGP Flowspec

This procedure disables BGP flowspec policy on an interface.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **{ ipv4 | ipv6 } flowspec disable**
4. **commit**

DETAILED STEPS

Step 1 **configure**

Step 2 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface GigabitEthernet 0/1/1/1
```

Configures an interface and enters the interface configuration mode.

Step 3 **{ ipv4 | ipv6 } flowspec disable**

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 flowspec disable
```

Disable flowspec policy on the selected interface.

Step 4 **commit**

Disable flowspec on the interface

The following example shows you how you can disable BGP flowspec on an interface, and apply another PBR policy:

```
Interface GigabitEthernet 0/0/0/0
 flowspec [ipv4/ipv6] disable
int g0/0/0/1
 service policy type pbr test_policy
!
```

Disable Flowspec Redirect and Validation

You can disable flowspec validation as a whole for eBGP sessions by means of configuring an explicit knob.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **address-family** { **ipv4** | **ipv6** }
5. **flowspec validation** { **disable** | **redirect disable** }

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0: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: RP/0/RP0/CPU0:router(config)# router bgp 100 neighbor 1.1.1.1	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 } Example: RP/0/RP0/CPU0:router(config-bgp)# router bgp 100 neighbor 1.1.1.1 address-family ipv4	Specifies either the IPv4 or IPv6 address family and enters address family configuration submenu, and initializes the global address family.

	Command or Action	Purpose
Step 5	flowspec validation { disable redirect disable } Example: <pre>RP/0/RP0/CPU0:router(config-bgp)# router bgp 100 neighbor 1.1.1.1 address-family ipv4 flowspec validation disable</pre>	You can choose to disable flowspec validation as a whole for all eBGP sessions or disable redirect nexthop validation.

Configuration Examples for Implementing BGP Flowspec

Flowspec Rule Configuration

Flowspec rule configuration example

In this example, two flowspec rules are created for two different VRFs with the goal that all packets to 10.0.1/24 from 192/8 and destination-port {range [137, 139] or 8080, rate limit to 500 bps in blue vrf and drop it in vrf-default. The goal is also to disable flowspec getting enabled on gig 0/0/0/0.

```
class-map type traffic match-all fs_tuple

match destination-address ipv4 10.0.1.0/24

match source-address ipv4 192.0.0.0/8

match destination-port 137-139 8080

end-class-map

!

!

policy-map type pbr fs_table_blue

class type traffic fs_tuple

police rate 500 bps

!

!

class class-default

!

end-policy-map

policy-map type pbr fs_table_default

class type traffic fs_tuple

drop

!
```

```

!
class class-default
!
end-policy-map
flowspec
local-install interface-all
address-family ipv4
  service-policy type pbr fs_table_default
!
!
vrf blue
  address-family ipv4
    service-policy type pbr fs_table_blue local
!
!
!
!
Interface GigabitEthernet 0/0/0/0
  vrf blue
  ipv4 flowspec disable

```

Drop Packet Length

This example shows a drop packet length action configuration:

```

class-map type traffic match-all match-pkt-len
  match packet length 100-150
end-class-map
!
policy-map type pbr test2
  class type traffic match-pkt-len
    drop
  !
  class type traffic class-default
  !
end-policy-map
!

```

To configure a traffic class to discard packets belonging to a specific class, you use the drop command in policy-map class configuration mode. In this example, a multi-range packet length value from 100-150 has been defined. If the packet length of the incoming traffic matches this condition, the action is defined to 'drop' this packet.

Redirect traffic and rate-limit: Example

```
class-map type traffic match-all match-src-ipv6-addr
  match source-address ipv6 3110:1::/48
end-class-map
!
policy-map type pbr test5
  class type traffic match-src-ipv6-addr
    redirect nexthop 3010:10:11::
    police rate 20 mbps
  !
  !
  class type traffic class-default
  !
end-policy-map
!
```

In this example, an action is defined in the flowspec rule to redirect all the traffic from a particular source P address (3110:1::/48) to a next hop address. Also, for any traffic that comes with this source-address, rate limit the source address to 20 megabits per second.

Redirect Traffic from Global to VRF (vrf1)

This example shows you the configuration for redirecting traffic from a global traffic link to an individual VRF interface.

```
class-map type traffic match-all match-src-ipv6-addr
  match source-address ipv6 3110:1::/48
end-class-map
!
policy-map type pbr test4
  class type traffic match-src-ipv6-addr
    redirect nexthop route-target 100:1
  !
  class type traffic class-default
  !
end-policy-map
```

Remark DSCP

This is an example of the set dscp action configuration.

```
class-map type traffic match-all match-dscp-af11
  match dscp 10
end-class-map
!
policy-map type pbr test6
  class type traffic match-dscp-af11
    set dscp af23
  !
  class type traffic class-default
  !
end-policy-map
!
```

In this example, the traffic marking extended community (**match dscp**) instructs the system to modify or set the DSCP bits of a transiting IP packet from dscp 10 to dscp af23.

Additional References for BGP Flowspec

The following sections provide references related to implementing BGP Flowspec.

Related Documents

Related Topic	Document Title
BGP flowspec commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Routing Command Reference for Cisco CRS Routers</i>

Standards

Standards	Title
draft-ietf-idr-flow-spec-v6-05	<i>Dissemination of Flow Specification Rules for IPv6</i> ,
draft-ietf-idr-flowspec-redirect-ip-01	BGP Flow-Spec Redirect to IP Action
draft-simpson-idr-flowspec-redirect-02	BGP Flow-Spec Extended Community for Traffic Redirect to IP Next Hop
draft-ietf-idr-bgp-flowspec-oid-02	Revised Validation Procedure for BGP Flow Specifications

RFCs

RFCs	Title
RFC 5575	Dissemination of Flow Specification Rules

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport



CHAPTER 4

Implementing BFD

This module describes the configuration of bidirectional forwarding detection (BFD) on the Cisco CRS Router.

Bidirectional forwarding detection (BFD) provides low-overhead, short-duration detection of failures in the path between adjacent forwarding engines. BFD allows a single mechanism to be used for failure detection over any media and at any protocol layer, with a wide range of detection times and overhead. The fast detection of failures provides immediate reaction to failure in the event of a failed link or neighbor.

Feature History for Implementing Bidirectional Forwarding Detection

Release	Modification
Release 3.2	<p>This feature was introduced with support for the following features:</p> <ul style="list-style-type: none">• IPv4 asynchronous and echo modes over physical POS and Gigabit Ethernet numbered links and VLANs.• BFD IPv4 single-hop.• Distribution on line cards.• BFD Version 0 and Version 1.
Release 3.3.0	<ul style="list-style-type: none">• Support was added to BFD for the following features:<ul style="list-style-type: none">• BFD over bundled VLANs using static routes.• Minimum disruption restart (MDR), which allows for a node CPU restart while minimizing traffic loss and network churn.• Fast reroute/Traffic engineering (FRR/TE) using BFD on Ethernet interfaces.• Configuration procedure was added to support the clear bfd counters packet and show bfd counters packet commands.

Release 3.3.2	<ul style="list-style-type: none"> • The echo disable command was added to enable users to disable echo mode on routers or interfaces where BFD is used in conjunction with Unicast Reverse Path Forwarding (uRPF). • A new BFD configuration mode was added, under which users can disable echo mode. The bfd command was added to allow users to enter the new BFD configuration mode.
Release 3.4.0	OSPF and IS-IS were supported on BFD over bundle VLANs.
Release 3.7.0	BFD for IPv6 was added on the Cisco CRS-1 Router.
Release 3.8.0	OSPFv3 over BFD support was added on the Cisco CRS-1 Router. BFD MIB support was added on the Cisco CRS-1 Router .
Release 3.9.0	<ul style="list-style-type: none"> • Support for these applications with BFD was added: <ul style="list-style-type: none"> • Hot Standby Router Protocol (HSRP) • Virtual Router Redundancy Protocol (VRRP) • The dampening command was added to minimize BFD session flapping and delay session startup. • The echo ipv4 source command was added to specify a source IP address and override the default. • The ipv6 checksum command was added to enable and disable the IPv6 UDP checksum computation and BFD interface configuration modes.

Release 4.0.0	<p>Support for these BFD features was added:</p> <ul style="list-style-type: none">• Support for BFD Per Member Links on Link Bundles was introduced.• Support for BFD was added on the following physical layer interface modules (PLIMs):<ul style="list-style-type: none">• 14-Port 10-Gigabit Ethernet LAN/WAN-PHY PLIM (14X10GBE-WL-XFP) (with the Cisco CRS-3 Modular Services Card or Cisco CRS-3 Forwarding Processor Card)• 20-Port 10-Gigabit Ethernet LAN/WAN-PHY PLIM (20X10GBE-WL-XFP) (with the Cisco CRS-3 Modular Services Card or Cisco CRS-3 Forwarding Processor Card)
---------------	---

Release 4.0.1	<p>Support for these BFD features was added:</p> <ul style="list-style-type: none"> • BFD on bundle member links was added for the following PLIMs: <ul style="list-style-type: none"> • 14-Port 10-Gigabit Ethernet LAN/WAN-PHY PLIM (14X10GBE-WL-XFP) (with the Cisco CRS-3 Modular Services Card or Cisco CRS-3 Forwarding Processor Card) • 20-Port 10-Gigabit Ethernet LAN/WAN-PHY PLIM (20X10GBE-WL-XFP) (with the Cisco CRS-3 Modular Services Card or Cisco CRS-3 Forwarding Processor Card) • 1-Port 100-Gigabit Ethernet PLIM (1X100GBE) (with the Cisco CRS-3 Modular Services Card or Cisco CRS-3 Forwarding Processor Card) • 4-Port 10-Gigabit Ethernet LAN/WAN-PHY PLIM (4-10GBE-WL-XFP) • 8-Port 10-Gigabit Ethernet LAN/WAN-PHY PLIM (8-10GBE-WL-XFP) • The echo latency detect command was added to enable latency detection for BFD echo packets on non-bundle interfaces. • The echo startup validate command was added to verify the echo path before starting a BFD session on non-bundle interfaces. • The trap singlehop pre-mapped command was added so that additional information can be provided along with the MIB trap regular information.
Release 4.2.1	<p>Support for these BFD features was added:</p> <ul style="list-style-type: none"> • Support for BFD over GRE, and Pseudowire headend was added. • The multihop ttl-drop-threshold command was added to specify the TTL value to start dropping packets for multihop sessions. • The multipath include command was added to specify the list of nodes eligible to host the multipath sessions.

Release 4.2.3	Support for BFD over Logical Bundle feature was added.
Release 4.3.0	Support for these features was added: <ul style="list-style-type: none"> • BFD over Logical Bundle
Release 4.3.1	Support for these features was added: <ul style="list-style-type: none"> • BFD IPv6 Multihop • BFD over MPLS Traffic Engineering LSPs • BFD over Pseudowire Head-end • BFD over Satellite Interfaces
Release 5.2.4	Support for BFD over Bundles CISCO/IETF mode support on a per bundle basis was added.

- [Prerequisites for Implementing BFD, on page 247](#)
- [Restrictions for Implementing BFD, on page 248](#)
- [Information About BFD, on page 249](#)
- [How to Configure BFD, on page 266](#)
- [Configuration Examples for Configuring BFD, on page 296](#)
- [Where to Go Next, on page 306](#)
- [Additional References, on page 306](#)

Prerequisites for Implementing BFD

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 following prerequisites are required to implement BFD:

- If enabling BFD on Multiprotocol Label Switching (MPLS), an installed composite PIE file including the MPLS package, or a composite-package image is required. For Border Gateway Protocol (BGP), Intermediate System-to-Intermediate System (IS-IS), Static, and Open Shortest Path First (OSPF), an installed Cisco IOS XR IP Unicast Routing Core Bundle image is required.
- Interior Gateway Protocol (IGP) is activated on the router if you are using IS-IS or OSPF.
- To enable BFD for a neighbor, the neighbor router must support BFD.
- In Cisco IOS XR releases before Release 3.9.0, we recommended that you configure the local router ID with the **router-id** command in global configuration mode prior to setting up a BFD session. If you did not configure the local router ID, then by default the source address of the IP packet for BFD echo mode is the IP address of the output interface. Beginning in Cisco IOS XR release 3.9.0 and later, you can use the **echo ipv4 source** command to specify the IP address that you want to use as the source address.
- To support BFD on bundle member links, be sure that the following requirements are met:

- The routers on either end of the bundle are connected back-to-back without a Layer 2 switch in between.
- For a BFD session to start, any one of the following configurations or states are present on the bundle member:
Link Aggregation Control Protocol (LACP) Distributing state is reached, –Or–
EtherChannel or POS Channel is configured, –Or–
Hot Standby and LACP Collecting state is reached.

Restrictions for Implementing BFD

These restrictions apply to BFD:

- Demand mode is not supported in Cisco IOS XR software.
- BFD echo mode is not supported for these features:
 - BFD for IPv4 on bundled VLANs
 - BFD for IPv6 (global and link-local addressing)
 - BFD with uRPF (IPv4 or IPv6)
 - Rack reload and online insertion and removal (OIR) when a BFD bundle interface has member links that span multiple racks
 - BFD for Multihop Paths
 - BFD over PWHE
 - BFD over GRE
- BFD for IPv6 has these restrictions:
 - BFD for IPv6 is not supported on bundled VLAN interfaces
 - BFD for IPv6 static routes that have link-local address as the next-hop is not supported
 - BFD for Multihop Paths is not supported
 - BFD over GRE is not supported
- For BFD on bundle member links, only a single BFD session for each bundle member link is created, monitored, and maintained for the IPv4 addressing type only. IPv6 and VLAN links in a bundle have the following restrictions:
 - IPv6 states are not explicitly monitored on a bundle member and they inherit the state of the IPv4 BFD session for that member interface.
 - VLAN subinterfaces on a bundle member also inherit the BFD state from the IPv4 BFD session for that member interface. VLAN subinterfaces are not explicitly monitored on a bundle member.
- Echo latency detection and echo validation are not supported on bundle interfaces.

- Only BGP application is supported on BFD for Multihop Paths.
- Only static and BGP applications are supported on BFD over PWHE.
- Only static, OSPF, IS-IS, and BGP applications are supported on BFD over GRE.

Information About BFD

Differences in BFD in Cisco IOS XR Software and Cisco IOS Software

If you are already familiar with BFD configuration in Cisco IOS software, be sure to consider the following differences in BFD configuration in the Cisco IOS XR software implementation:

- In Cisco IOS XR software, BFD is an application that is configured under a dynamic routing protocol, such as an OSPF or BGP instance. This is not the case for BFD in Cisco IOS software, where BFD is only configured on an interface.
- In Cisco IOS XR software, a BFD neighbor is established through routing. The Cisco IOS **bfd neighbor** interface configuration command is not supported in Cisco IOS XR software.
- Instead of using a dynamic routing protocol to establish a BFD neighbor, you can establish a specific BFD peer or neighbor for BFD responses in Cisco IOS XR software using a method of static routing to define that path. In fact, you must configure a static route for BFD if you do not configure BFD under a dynamic routing protocol in Cisco IOS XR software.
- A router running BFD in Cisco IOS software can designate a router running BFD in Cisco IOS XR software as its peer using the **bfd neighbor** command; the Cisco IOS XR router must use dynamic routing or a static route back to the Cisco IOS router to establish the peer relationship. See the [BFD Peers on Routers Running Cisco IOS and Cisco IOS XR Software: Example](#).

BFD Modes of Operation

Cisco IOS XR software supports the asynchronous mode of operation only, with or without using echo packets. Asynchronous mode without echo will engage various pieces of packet switching paths on local and remote systems. However, asynchronous mode with echo is usually known to provide slightly wider test coverage as echo packets are self-destined packets which traverse same packet switching paths as normal traffic on the remote system.

BFD echo mode is enabled by default for the following interfaces:

- For IPv4 on member links of BFD bundle interfaces.
- For IPv4 on other physical interfaces whose minimum interval is less than two seconds.

When BFD is running asynchronously without echo packets ([Figure 35](#)), the following occurs:

- Each system periodically sends BFD control packets to one another. Packets sent by BFD router “Peer A” to BFD router “Peer B” have a source address from Peer A and a destination address for Peer B.
- Control packet streams are independent of each other and do not work in a request/response model.
- If a number of packets in a row are not received by the other system, the session is declared down.

Figure 14: BFD Asynchronous Mode Without Echo Packets

When BFD is running asynchronously with echo packets (Figure 36), the following occurs:

- BFD echo packets are looped back through the forwarding path only of the BFD peer and are not processed by any protocol stack. So, packets sent by BFD router “Peer A” can be sent with both the source and destination address of Peer A.
- BFD echo packets are sent in addition to BFD control packets.

Figure 15: BFD Asynchronous Mode With Echo Packets

For more information about control and echo packet intervals in asynchronous mode, see the [BFD Packet Intervals and Failure Detection](#).

BFD Packet Information

BFD Source and Destination Ports

BFD payload control packets are encapsulated in UDP packets, using destination port 3784 and source port 49152. Even on shared media, like Ethernet, BFD control packets are always sent as unicast packets to the BFD peer.

Echo packets are encapsulated in UDP packets, as well, using destination port 3785 and source port 3785.

The BFD over bundle member feature increments each byte of the UDP source port on echo packets with each transmission. UDP source port ranges from 0xC0C0 to 0xFFFF. For example:

1st echo packet: 0xC0C0

2nd echo packet: 0xC1C1

3rd echo packet: 0xC2C2

The UDP source port is incremented so that sequential echo packets are hashed to deviating bundle member.

BFD Packet Intervals and Failure Detection

BFD uses configurable intervals and multipliers to specify the periods at which control and echo packets are sent in asynchronous mode and their corresponding failure detection.

There are differences in how these intervals and failure detection times are implemented for BFD sessions running over physical interfaces, and BFD sessions on bundle member links.

BFD Packet Intervals on Physical Interfaces

When BFD is running over physical interfaces, echo mode is used only if the configured interval is less than two seconds.

BFD sessions running over physical interfaces when echo mode is enabled send BFD control packets at a slow rate of every two seconds. There is no need to duplicate control packet failure detection at a fast rate because BFD echo packets are already being sent at fast rates and link failures will be detected when echo packets are not received within the echo failure detection time.

BFD Packet Intervals on Bundle Member Links

On each bundle member interface, BFD asynchronous mode control packets run at user-configurable interval and multiplier values, even when echo mode is running.

However, on a bundle member interface when echo mode is enabled, BFD asynchronous mode must continue to run at a fast rate because one of the requirements of enabling BFD echo mode is that the bundle member interface is available in BFD asynchronous mode.

The maximum echo packet interval for BFD on bundle member links is the minimum of either 30 seconds or the asynchronous control packet failure detection time.

When echo mode is disabled, the behavior is the same as BFD over physical interfaces, where sessions exchange BFD control packets at the configured rate.

Control Packet Failure Detection In Asynchronous Mode

Control packet failure in asynchronous mode without echo is detected using the values of the minimum interval (`bfd minimum-interval` for non-bundle interfaces, and `bfd address-family ipv4 minimum-interval` for bundle interfaces) and multiplier (`bfd multiplier` for non-bundle interfaces, and `bfd address-family ipv4 multiplier` for bundle interfaces) commands.

For control packet failure detection, the local multiplier value is sent to the neighbor. A failure detection timer is started based on $(I \times M)$, where I is the negotiated interval, and M is the multiplier provided by the remote end.

Whenever a valid control packet is received from the neighbor, the failure detection timer is reset. If a valid control packet is not received from the neighbor within the time period $(I \times M)$, then the failure detection timer is triggered, and the neighbor is declared down.

Echo Packet Failure Detection In Asynchronous Mode

The standard echo failure detection scheme is done through a counter that is based on the value of the **bfd multiplier** command on non-bundle interfaces, and the value of the **bfd address-family ipv4 multiplier** command for bundle interfaces.

This counter is incremented each time the system sends an echo packet, and is reset to zero whenever *any* echo packet is received, regardless of the order that the packet was sent in the echo packet stream.

Under ideal conditions, this means that BFD generally detects echo failures that exceed the period of time $(I \times M)$ or $(I \times M \times M)$ for bundle interfaces, where:

- I —Value of the minimum interval (`bfd minimum-interval` for non-bundle interfaces, and **bfd address-family ipv4 minimum-interval** for bundle interfaces).
- M —Value of the multiplier (**bfd multiplier** for non-bundle interfaces, and **bfd address-family ipv4 multiplier** for bundle interfaces) commands.

So, if the system transmits one additional echo packet beyond the multiplier count without receipt of any echo packets, echo failure is detected and the neighbor is declared down (See [Example 2](#)).

However, this standard echo failure detection does not address latency between transmission and receipt of any specific echo packet, which can build beyond $(I \times M)$ over the course of the BFD session. In this case, BFD will not declare a neighbor down as long as any echo packet continues to be received within the multiplier window and resets the counter to zero. Beginning in Cisco IOS XR 4.0.1, you can configure BFD to measure this latency for non-bundle interfaces. For more information, see [Example 3](#) and the [Echo Packet Latency](#).

Echo Failure Detection Examples

This section provides examples of several scenarios of standard echo packet processing and failure detection without configuration of latency detection for non-bundle interfaces. In these examples, consider an interval of 50 ms and a multiplier of 3.



Note The same interval and multiplier counter scheme for echo failure detection is used for bundle interfaces, but the values are determined by the **bfd address-family ipv4 multiplier** and **bfd address-family ipv4 minimum-interval** commands, and use a window of $(I \times M \times M)$ to detect absence of receipt of echo packets.

Example 1

The following example shows an ideal case where each echo packet is returned before the next echo is transmitted. In this case, the counter increments to 1 and is returned to 0 before the next echo is sent and no echo failure occurs. As long as the roundtrip delay for echo packets in the session is less than the minimum interval, this scenario occurs:

```
Time (T): Echo#1 TX (count = 1)
T + 1 ms: Echo#1 RX (count = 0)
T + 50 ms: Echo#2 TX (count = 1)
T + 51 ms: Echo#2 RX (count = 0)
T + 100 ms: Echo#3 TX (count = 1)
T + 101 ms: Echo#3 RX (count = 0)
T + 150 ms: Echo#4 TX (count = 1)
T + 151 ms: Echo#4 RX (count = 0)
```

Example 2

The following example shows the absence in return of any echo packets. After the transmission of the fourth echo packet, the counter exceeds the multiplier value of 3 and echo failure is detected. In this case, echo failure detection occurs at the 150 ms $(I \times M)$ window:

```
Time (T): Echo#1 TX (count = 1)
T + 50 ms: Echo#2 TX (count = 2)
T + 100 ms: Echo#3 TX (count = 3)
T + 150 ms: Echo#4 TX (count = 4 -> echo failure)
```

Example 3

The following example shows an example of how roundtrip latency can build beyond $(I \times M)$ for any particular echo packet over the course of a BFD session using the standard echo failure detection, but latency between return of echo packets overall in the session never exceeds the $(I \times M)$ window and the counter never exceeds the multiplier, so the neighbor is not declared down.



Note You can configure BFD to detect roundtrip latency on non-bundle interfaces using the **echo latency detect** command beginning in Cisco IOS XR 4.0.1.

```
Time (T): Echo#1 TX (count = 1)
T + 1 ms: Echo#1 RX (count = 0)
T + 50 ms: Echo#2 TX (count = 1)
T + 51 ms: Echo#2 RX (count = 0)
T + 100 ms: Echo#3 TX (count = 1)
T + 150 ms: Echo#4 TX (count = 2)
T + 151 ms: Echo#3 RX (count = 0; ~50 ms roundtrip latency)
T + 200 ms: Echo#5 TX (count = 1)
T + 250 ms: Echo#6 TX (count = 2)
T + 251 ms: Echo#4 RX (count = 0; ~100 ms roundtrip latency)
T + 300 ms: Echo#7 TX (count = 1)
T + 350 ms: Echo#8 TX (count = 2)
T + 351 ms: Echo#5 RX (count = 0; ~150 ms roundtrip latency)
T + 451 ms: Echo#6 RX (count = 0; ~200 ms roundtrip latency; no failure detection)
T + 501 ms: Echo#7 RX (count = 0; ~200 ms roundtrip latency; no failure detection)
T + 551 ms: Echo#8 RX (count = 0; ~200 ms roundtrip latency; no failure detection)
```

Looking at the delay between receipt of echo packets for the BFD session, observe that no latency is beyond the ($I \times M$) window:

```
Echo#1 RX - Echo#2 RX: 50 ms
Echo#2 RX - Echo#3 RX: 100ms
Echo#3 RX - Echo#4 RX: 100ms
Echo#4 RX - Echo#5 RX: 100ms
Echo#5 RX - Echo#6 RX: 100ms
Echo#6 RX - Echo#7 RX: 50ms
Echo#7 RX - Echo#8 RX: 50ms
```

Summary of Packet Intervals and Failure Detection Times for BFD on Bundle Interfaces

For BFD on bundle interfaces, with a session interval I and a multiplier M , these packet intervals and failure detection times apply for BFD asynchronous mode ([Table 4: BFD Packet Intervals and Failure Detection Time Examples on Bundle Interfaces](#)):

- Value of I —Minimum period between sending of BFD control packets.
- Value of $I \times M$
 - BFD control packet failure detection time.
 - Minimum period between sending of BFD echo packets.

The BFD control packet failure detection time is the maximum amount of time that can elapse without receipt of a BFD control packet before the BFD session is declared down.

- Value of $(I \times M) \times M$ —BFD echo packet failure detection time. This is the maximum amount of time that can elapse without receipt of a BFD echo packet (using the standard multiplier counter scheme as described in [Echo Packet Failure Detection In Asynchronous Mode](#)) before the BFD session is declared down.

Table 4: BFD Packet Intervals and Failure Detection Time Examples on Bundle Interfaces

Configured Async Control Packet Interval (ms) (bfd address-family ipv4 minimum-interval)	Configured Multiplier (bfd address-family ipv4 multiplier)	Async Control Packet Failure Detection Time (ms) (Interval x Multiplier)	Echo Packet Interval (Async Control Packet Failure Detection Time)	Echo Packet Failure Detection Time (Echo Interval x Multiplier)
33	3	99	99	297
50	3	150	150	450
75	4	300	300	1200
200	2	400	400	800
2000	3	6000	6000	18000
15000	3	45000	30000 ¹	90000

¹ The maximum echo packet interval for BFD on bundle member links is the minimum of either 30 seconds or the asynchronous control packet failure detection time.

Echo Packet Latency

In Cisco IOS XR software releases prior to Cisco IOS XR 4.0.1, BFD only detects an absence of receipt of echo packets, not a specific delay for TX/RX of a particular echo packet. In some cases, receipt of BFD echo packets in general can be within their overall tolerances for failure detection and packet transmission, but a longer delay might develop over a period of time for any particular roundtrip of an echo packet (See [Example 3](#)).

Beginning in Cisco IOS XR Release 4.0.1, you can configure the router to detect the actual latency between transmitted and received echo packets on non-bundle interfaces and also take down the session when the latency exceeds configured thresholds for that roundtrip latency. For more information, see the [Configuring BFD Session Teardown Based on Echo Latency Detection](#).

In addition, you can verify that the echo packet path is within specified latency tolerances before starting a BFD session. With echo startup validation, an echo packet is periodically transmitted on the link while it is down to verify successful transmission within the configured latency before allowing the BFD session to change state. For more information, see the [Delaying BFD Session Startup Until Verification of Echo Path and Latency](#).

Priority Settings for BFD Packets

For all interfaces under over-subscription, the internal priority needs to be assigned to remote BFD Echo packets, so that these BFD packets are not overwhelmed by other data packets. In addition, CoS values need to be set appropriately, so that in the event of an intermediate switch, the reply back of remote BFD Echo packets are protected from all other packets in the switch.

As configured CoS values in ethernet headers may not be retained in Echo messages, CoS values must be explicitly configured in the appropriate egress QoS service policy. CoS values for BFD packets attached to a traffic class can be set using the set cos command. For more information on configuring class-based

unconditional packet marking, see “Configuring Modular QoS Packet Classification” in the *Modular QoS Configuration Guide for Cisco CRS Routers*.

BFD for IPv4

Cisco IOS XR software supports bidirectional forwarding detection (BFD) singlehop and multihop for both IPv4 and IPv6.

In BFD for IPv4 single-hop connectivity, Cisco IOS XR software supports both asynchronous mode and echo mode over physical numbered Packet-over-SONET/SDH (POS) and Gigabit Ethernet links, as follows:

- Echo mode is initiated only after a session is established using BFD control packets. Echo mode is always enabled for BFD bundle member interfaces. For physical interfaces, the BFD minimum interval must also be less than two seconds to support echo packets.
- BFD echo packets are transmitted over UDP/IPv4 using source and destination port 3785. The source address of the IP packet is the IP address of the output interface (default) or the address specified with the **router-id** command if set or the address specified in the **echo ipv4 source** command, and the destination address is the local interface address.
- BFD asynchronous packets are transmitted over UDP and IPv4 using source port 49152 and destination port 3784. For asynchronous mode, the source address of the IP packet is the local interface address, and the destination address is the remote interface address.



Note BFD multihop does not support echo mode.

Consider the following guidelines when configuring BFD on Cisco IOS XR software:

- BFD is a fixed-length hello protocol, in which each end of a connection transmits packets periodically over a forwarding path. Cisco IOS XR software supports BFD adaptive detection times.
- BFD can be used with the following applications:
 - BGP
 - IS-IS
 - EIGRP
 - OSPF
and OSPFv3
 - MPLS Traffic Engineering (MPLS-TE)
 - Static routes (IPv4 and IPv6)
 - Protocol Independent Multicast (PIM)
 - Hot Standby Router Protocol (HSRP)
 - Virtual Router Redundancy Protocol (VRRP)



Note When multiple applications share the same BFD session, the application with the most aggressive timer wins locally. Then, the result is negotiated with the peer router.

- BFD is supported for connections over the following interface types:
 - Gigabit Ethernet (GigE)
 - Hundred Gigabit Ethernet (HundredGigE)
 - Ten Gigabit Ethernet (TenGigE)
 - Packet-over-SONET/SDH (POS)
 - Serial
 - Virtual LAN (VLAN)
 - Satellite Interface
 - Logical interfaces such as bundles, GRE, PWHE



Note BFD is supported on the above interface types and not on logical interfaces unless specifically stated.

- Cisco IOS XR software supports BFD Version 0 and Version 1. BFD sessions are established using either version, depending upon the neighbor. BFD Version 1 is the default version and is tried initially for session creation.

BFD for IPv6

Cisco IOS XR software supports bidirectional forwarding detection (BFD) for both IPv4 and IPv6. Bidirectional forwarding detection (BFD) for IPv6 supports the verification of live connectivity on interfaces that use IPv6 addresses.

The live connectivity verification for both IPv4 and IPv6 interfaces is performed by the same services and processes. Both IPv4 and IPv6 BFD sessions can run simultaneously on the same line card.

The same features and configurations that are supported in BFD for IPv4 are also supported in BFD for IPv6, except for the following restrictions:

- BFD for IPv6 is not supported on bundled VLAN interfaces, GRE, and for Multiple paths.
- BFD for IPv6 is not supported in echo mode.

BFD on Bundled VLANs

BFD for IPv4 on bundled VLANs is supported using static routing, IS-IS, and OSPF. When running a BFD session on a bundled VLAN interface, the BFD session is active as long as the VLAN bundle is up.

As long as the VLAN bundle is active, the following events do not cause the BFD session to fail:

- Failure of a component link.
- Online insertion and removal (OIR) of a line card which hosts one or more of the component links.
- Addition of a component link (by configuration) to the bundle.
- Removal of a component link (by configuration) from the bundle.
- Shutdown of a component link.
- RP switchover.



Note For more information on configuring a VLAN bundle, see the *Configuring Link Bundling on Cisco IOS XR Software* module.

Keep the following in mind when configuring BFD over bundled VLANs:

- In the case of an RP switchover, configured next-hops are registered in the Routing Information Base (RIB).
- In the case of a BFD restart, static routes remain in the RIB. BFD sessions are reestablished when BFD restarts.

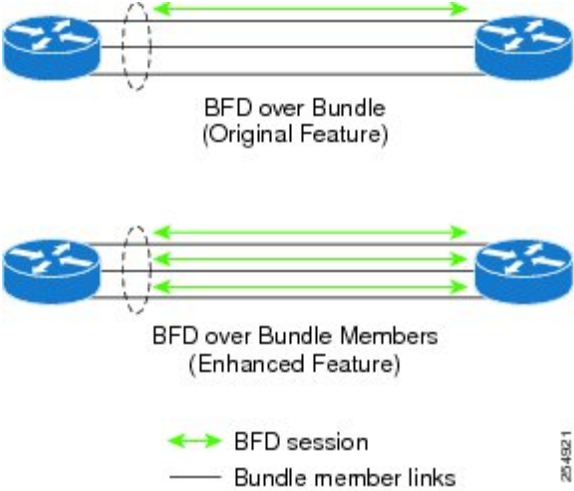


Note Static BFD sessions are supported on peers with address prefixes whose next-hops are directly connected to the router.

BFD Over Member Links on Link Bundles

Beginning in Cisco IOS XR Release 4.0, the BFD supports BFD sessions on individual physical bundle member links to monitor Layer 3 connectivity on those links, rather than just at a single bundle member as in prior releases ([Figure 37](#)).

Figure 16: BFD Sessions in Original BFD Over Bundles and Enhanced BFD Over Bundle Member Links Architectures



When you run BFD on link bundles, you can run an independent BFD session on each underlying physical interface that is part of that bundle.

When BFD is running on a link bundle member, these layers of connectivity are effectively tested as part of the interface state monitoring for BFD:

- Layer 1 physical state
- Layer 2 Link Access Control Protocol (LACP) state
- Layer 3 BFD state

The BFD agent on each bundle member link monitors state changes on the link. BFD agents for sessions running on bundle member links communicate with a bundle manager. The bundle manager determines the state of member links and the overall availability of the bundle. The state of the member links contributes to the overall state of the bundle based on the threshold of minimum active links or minimum active bandwidth that is configured for that bundle.

Overview of BFD State Change Behavior on Member Links and Bundle Status

This section describes when bundle member link states are characterized as active or down, and their effect on the overall bundle status:

- You can configure BFD on a bundle member interface that is already active or one that is inactive. For the BFD session to be *up* using LACP on the interface, LACP must have reached the *distributing* state. A BFD member link is “IIR Active” if the link is in LACP distributing state and the BFD session is up.
- A BFD member link is “IIR Attached” when the BFD session is down, unless a LACP state transition is received.
- You can configure timers for up to 3600 seconds (1 hour) to allow for delays in receipt of BFD state change notifications (SCNs) from peers before declaring a link bundle BFD session down. The configurable timers apply to these situations:
 - BFD session startup (**bfd address-family ipv4 timers start** command)—Number of seconds to allow after startup of a BFD member link session for the expected notification from the BFD peer

to be received to declare the session up. If the SCN is not received after that period of time, the BFD session is declared down.

- Notification of removal of BFD configuration by a neighbor (**bfd address-family ipv4 timers nbr-unconfig** command)—Number of seconds to allow after receipt of notification that BFD configuration has been removed by a BFD neighbor so that any configuration inconsistency between the BFD peers can be fixed. If the BFD configuration issue is not resolved before the specified timer is reached, the BFD session is declared down.
- A BFD session sends a DOWN notification when one of these occurs:
 - The BFD configuration is removed on the local member link.

The BFD system notifies the peer on the neighbor router that the configuration is removed. The BFD session is removed from the bundle manager without affecting other bundle member interfaces or the overall bundle state.
 - A member link is removed from the bundle.

Removing a member link from a bundle causes the bundle member to be removed ungracefully. The BFD session is deleted and BFD on the neighboring router marks the session DOWN rather than NBR_CONFIG_DOWN.
- In these cases, a DOWN notification is not sent, but the internal infrastructure treats the event as if a DOWN has occurred:
 - The BFD configuration is removed on a neighboring router and the neighbor unconfiguration timer (if configured) expires.

The BFD system notifies the bundle manager that the BFD configuration has been removed on the neighboring router and, if **bfd timers nbr-unconfig** is configured on the link, the timer is started. If the BFD configuration is removed on the local router before the timer expires, then the timer is stopped and the behavior is as expected for BFD configuration removal on the local router.

If the timer expires, then the behavior is the same as for a BFD session DOWN notification.
 - The session startup timer expires before notification from the BFD peer is received.
- The BFD session on a bundle member sends BFD state change notifications to the bundle manager. Once BFD state change notifications for bundle member interfaces are received by the bundle manager, the bundle manager determines whether or not the corresponding bundle interface is usable.
- A threshold for the minimum number of active member links on a bundle is used by the bundle manager to determine whether the bundle remains active, or is down based on the state of its member links. When BFD is started on a bundle that is already active, the BFD state of the bundle is declared when the BFD state of all the existing active members is known.

Whenever a member's state changes, the bundle manager determines if the number of active members is less than the minimum number of active links threshold. If so, then the bundle is placed, or remains, in DOWN state. Once the number of active links reaches the minimum threshold then the bundle returns to UP state.
- Another threshold is configurable on the bundle and is used by the bundle manager to determine the minimum amount of active bandwidth to be available before the bundle goes to DOWN state. This is configured using the **bundle minimum-active bandwidth** command.

- The BFD server responds to information from the bundle manager about state changes for the bundle interface and notifies applications on that interface while also sending system messages and MIB traps.

BFD Multipath Sessions

BFD can be applied over virtual interfaces such as GRE tunnel interfaces, PWHE interfaces, or between interfaces that are multihops away as described in the [BFD for MultiHop Paths](#) section. These types of BFD sessions are referred to BFD Multipath sessions.

As long as one path to the destination is active, these events may or may not cause the BFD Multipath session to fail as it depends on the interval negotiated versus the convergence time taken to update forwarding plane:

- Failure of a path
- Online insertion or removal (OIR) of a line card which hosts one or more paths
- Removal of a link (by configuration) which constitutes a path
- Shutdown of a link which constitutes a path

You must configure **bfd multipath include location** *location-id* command to enable at least one line card for the underlying mechanism that can be used to send and receive packets for the multipath sessions.

If a BFD Multipath session is hosted on a line card that is being removed from the `bfd multipath include` configuration, online removed, or brought to maintenance mode, then BFD attempts to migrate all BFD Multipath sessions hosted on that line card to another one. In that case, static routes are removed from RIB and then the BFD session is established again and included to RIB.

For more information on PW headend and its configuration, see *Implementing Virtual Private LAN Services* module in the *Virtual Private Network Configuration Guide for Cisco CRS Routers*. For more information on GRE, see *Implementing MPLS Layer 2 VPNs* module in *Virtual Private Network Configuration Guide for Cisco CRS Routers*



Note BFD runs at a faster pace than the GRE keepalive. The configuration to enable or disable BFD to run over a GRE tunnel is in the BGP and/or IGP domain.

BFD for MultiHop Paths

BFD multihop (BFD-MH) is a BFD session between two addresses that are not on the same subnet. An example of BFD-MH is a BFD session between PE and CE loopback addresses or BFD sessions between routers that are several TTL hops away. The applications that support BFD multihop are external and internal BGP. BFD multihop supports BFD on arbitrary paths, which can span multiple network hops.

The BFD Multihop feature provides sub-second forwarding failure detection for a destination more than one hop, and up to 255 hops, away. The **bfd multihop ttl-drop-threshold** command can be used to drop BFD packets coming from neighbors exceeding a certain number of hops. BFD multihop is supported on all currently supported media-type for BFD singlehop.

Setting up BFD Multihop

A BFD multihop session is set up between a unique source-destination address pair provided by the client. A session can be set up between two endpoints that have IP connectivity. For BFD Multihop, IPv4 addresses in both global routing table and in a VRF is supported.

When BFD is used with BGP, the BFD session type (singlehop or multihop) is configured based on the BGP configuration. If you configure eBGP-multihop keyword, the BFD session will also run in multihop mode; otherwise the session will run in singlehop mode.

Limitations of BFD

These are some performance limitations of BFD for Multihop Paths, GRE, and Pseudowire:

- In the case of Taiko linecards, 50msec or longer interval for each BFD session and maximum of 7000 pps and the maximum number of sessions can be 1250.
- In the case of Metro linecards, 100msec or longer interval for each BFD session and maximum of 4000pps and the maximum number of sessions can be 1250.

BFD over MPLS Traffic Engineering LSPs

Bidirectional Forwarding Detection (BFD) over MPLS Traffic Engineering Label Switched Paths (LSPs) feature in Cisco IOS XR Software detects MPLS Label Switched Path LSP data plane failures. Since the control plane processing required for BFD control packets is relatively smaller than the processing required for LSP Ping messages, BFD can be deployed for faster detection of data plane failure for a large number of LSPs.

The BFD over MPLS TE LSPs implementation in Cisco IOS XR Software is based on *RFC 5884: Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)*. LSP Ping is an existing mechanism for detecting MPLS data plane failures and for verifying the MPLS LSP data plane against the control plane. BFD can be used for detecting MPLS data plane failures, but not for verifying the MPLS LSP data plane against the control plane. A combination of LSP Ping and BFD provides faster data plane failure detection on a large number of LSPs.

The BFD over MPLS TE LSPs is used for networks that have deployed MPLS as the multi service transport and that use BFD as fast failure detection mechanism to enhance network reliability and up time by using BFD as fast failure detection traffic black holing.

BFD over MPLS TE LSPs support:

- BFD async mode (BFD echo mode is not supported)
- IPv4 only, since MPLS core is IPv4
- BFD packets will carry IP DSCP 6 (Internet Control)
- Use of BFD for TE tunnel bring up, re-optimization, and path protection (Standby and FRR)
- Fastest detection time (100 ms x 3 = 300 ms)
- Optional Periodic LSP ping verification after BFD session is up
- Dampening to hold-down BFD failed path-option
- There are two ways in which the BFD packets from tail-end to head-end will be used:

- BFD packets from tail-end to head-end will be IP routed (IPv4 Multihop - port# 4784)
- BFD packets from tail-end to head-end will be Label Switched (port# 3784) if MPLS LDP is available in Core with label path from tail-end to head-end.

Echo Timer configuration for BFD on Bundle Interfaces

The echo timer configuration allows you to specify the minimum interval for echo packets on IPv4 BFD sessions on bundle member links. You can set the echo timer value globally using the **bfd echo ipv4 bundle-per-member minimum-interval** command. Use the **bfd address-family ipv4 echo minimum-interval** to locally set the minimum interval value for the bundle ethernet interface.



Note This feature is applicable only for Cisco standard BFD over bundle per-member link mode.

See the *BFD Commands on Cisco IOS XR Routing Command Reference for the Cisco CRS Router* guide for details on these commands.

The echo timer behavior with the global and local echo configuration combination is illustrated in the following table:

Table 5: Echo timer behavior with global and local echo configuration

Global echo min-interval value Command: bfd echo ipv4 bundle-per-member minimum-interval	Local bundle ethernet interface specific echo min-interval value Command: bfd address-family ipv4 echo minimum-interval	Echo value used by the BoB session based on global and local configurations
Not configured	Not Configured	Async * multiplier
Global value is lesser than Async * multiplier	Not Configured	Async * multiplier
Global value is greater than Async * multiplier	Not Configured	Global
Not configured	Local is greater than Async * Multiplier	Local
Not configured	Local is lesser than Async * Multiplier	Async * multiplier
Global is configured (any value)	Local is greater than Async * Multiplier	Local
Global is configured (any value)	Local is lesser than Async * Multiplier	Async * multiplier

**Note**

- Multiplier in the table refers to the remote multiplier value.
- Async refers to the negotiated asynchronous minimum interval value.

When R5.3.0 devices have BoB sessions with devices running on versions lesser than R5.3.0, it is recommended to retain the default echo timer value or configure identical values on both the devices.

Bidirectional Forwarding Detection over Logical Bundle

The Bidirectional Forwarding Detection (BFD) over Logical Bundle feature implements and deploys BFD over bundle interfaces based on RFC 5880. The BFD over Logical Bundle (BLB) feature replaces the BVLAN feature and resolves certain interoperability issues with other platforms that run BFD over bundle interface in pure RFC5880 fashion. These platforms include products of other vendors, as well as other Cisco products running Cisco IOS or Cisco Nexus OS software.

BLB is a multipath (MP) single-hop session. BLB requires limited knowledge of the bundle interfaces on which the sessions run; this is because BFD treats the bundle as one big pipe. To function, BLB requires only information about IP addresses, interface types, and caps on bundle interfaces. Information such as list of bundle members, member states, and configured minimum or maximum bundle links are not required.

BLB is supported on IPv4 address, IPv6 global address, and IPv6 link-local address. The current version of the software supports a total of 200 sessions (which includes BFD Single hop for physical and logical sub-interfaces; BFD over Bundle (BoB) and BLB) per line card. The maximum processing capability of BFD control packets, per line card, has also increased to 7000 pps (packets per second).

**Note**

ISSU is not supported for BFD over Logical Bundle feature.

BFD IPv6 Multihop

Bidirectional Forwarding Detection (BFD) Multihop IPv6 (MHv6) feature supports BFD sessions between interfaces that are multiple hops away. The BFD MHv6 enables a BFD session between two addresses (BFD session between provider edge (PE) and customer edge (CE) loopback addresses or BFD session between routers that are several time-to-live (TTL) hops away) that are not on the same interface. BFD MHv6 is supported in a typical CE – PE configuration over loopback as well as the physical interface addresses, with static IPv6 routes using iBGP/eBGP as the client application. BFD Multihop provides continuity check (CC) on arbitrary paths spanning multiple network hops and provides failure notifications for Multihop protocols like BGP (iBGP: VRF and non-VRF and eBGP: VRF and non-VRF), MPLS Traffic Engineering, and LDP. The Cisco IOS XR Software BFD MHv6 implementation is in accordance with *IETF RFC5883 for IPv6 networks*.

BFD support for IPv6 Multihop is on a par with the BFD support for IPv4 Multihop (BFD MHv4). BFD MHv6 is supported on Cisco CRS-3 Modular Services line card and Cisco CRS Modular Services Card line card. The minimum interval for BFD Multihop is 50 milliseconds on Cisco CRS-3 Modular Services line card and 100 milliseconds on Cisco CRS Modular Services line card.



Note BFD over 6VPE/6PE is not supported. The BFD MHv6 does not support BFD echo mode.

BFD IPv6 Multihop removes the restriction of a single path IPv6 BFD session, where the BFD neighbor is always one hop away, and the BFD Agent in the line card always receives or transmits BFD packets over a local interface on the same line card.

The BFD switching mechanism for IPv6 Multihop link is employed when the BFD packets are transmitted from one end point node to the other. The BFD punting mechanism is employed when BFD packets are received at the remote end point node.

BFD over Pseudowire Headend

The Bidirectional Forwarding Detection over Pseudowire Headend (BFD_oPWHE) feature enables BFD support over the customer edge (CE) to pseudowire headend (S-PE) links for fast failure detection along the path between the eBGP neighbors.

BFD over PWHE supports:

- BFD sessions per pseudo-wire for end-to-end fault detection between the CE and PWHE PE
- BFDv4 for IPv4 and BFDv6 for IPv6 (static and BGP)
- BFD asynchronous mode over PWHE
- Pseudowire VC type 4 and type 5

For PWHE to be operational, the BFD agent should be hosted on one of the line cards that is part of the PWHE generic interface list. The BFD multipath must be configured for a line card that is part of the generic interfaces list.

Use the **bfd multipath include location** *node-id* command to include specific line cards to host BFD multiple path sessions and thereby enable BFD over PWHE.

BFD over Satellite Interfaces

Bidirectional Forwarding Detection (BFD) over satellite interfaces feature enables BFD support on satellite line cards. Satellite interfaces are known as virtual (bundle) interfaces. BFD uses multipath infrastructure to support BFD on satellite line cards. BFD over satellite is a multipath (MP) single-hop session and is supported on IPv4 address, IPv6 global address, and IPv6 link-local address. BFD over Satellite is supported on the Cisco CRS-3 Modular Services Line Card or the Cisco CRS Modular Services Line Card.

Limitations

These limitations apply for BFD over Satellite interfaces:

- BFD async mode is supported on Satellite GigabitEthernet links when they are not part of the bundle.
- BFD echo mode is not supported on Satellite GigabitEthernet links.
- BFD over bundles (BOB) is not supported.
- When the Satellite links are part of the Access Bundle, only BFD over Logical Bundle (BLB) is supported.

- In BLB, only one BFD session is supported on the bundle.
- The nV Satellite access port bundles does not support BFD over bundles (BoB) when the Inter-Chassis Link (ICL) is also a bundle (Bundle over Bundle topology).

BFD over Bundles CISCO/IETF Mode Support on a Per Bundle Basis

BFD over Bundle (BoB) mode is a standard based fast failure detection of link aggregation (LAG) member links that is interoperable between different platforms. BoB support on a per bundle basis provides an option to choose either Cisco or IETF standard per bundle, without necessitating reloads or process restarts across various systems. The default is Cisco mode.



Note The global-level command available in previous releases to configure CISCO/IETF BoB over bundles is deprecated from release 5.3.1 onwards. In order to ensure a smooth upgrade, Cisco recommends that you configure the bundle at the interface level.

- The Cisco mode uses CDP MAC whereas IETF mode uses IANA assigned MAC.
- Cisco BFD over Bundle sessions use destination UDP port: 3784, while IETF BFD over Bundle sessions use destination UDP port: 6784.

Restrictions

These limitations apply for the BFD over Bundle Mode feature:

- The BFD mode change (Cisco to IETF and vice-versa) goes through only when the BFD state for the bundle is 'down' or 'BoB nonoperational.'



Note You can use the **no bfd address-family ipv4 fast-detect** command to make BoB non-operational. You can also choose to configure a bundle to 'down' state by configuring shutdown under that particular bundle.

- For a bundle to accept the new BFD mode change, you must bring down and then recreate the existing BFD sessions.
- BFD Echo mode is not supported in IETF BFD over Bundle (BoB) sessions.

BFD Object Tracking

Object Tracking is enhanced to support BFD to track the reachability of remote IP addresses. This will enable complete detection and HSRP switch over to happen within a time of less than one second as BFD can perform the detection in the order of few milliseconds

How to Configure BFD

BFD Configuration Guidelines

Before you configure BFD, consider the following guidelines:

- FRR/TE, FRR/IP, and FRR/LDP using BFD is supported on POS interfaces and Ethernet interfaces.
- To establish a BFD neighbor in Cisco IOS XR software, BFD must either be configured under a dynamic routing protocol, or using a static route.
- The maximum rate in packets-per-second (pps) for BFD sessions is linecard-dependent. If you have multiple linecards supporting BFD, then the maximum rate for BFD sessions per system is the supported linecard rate multiplied by the number of linecards.

To know the BFD scale values, use the **show bfd summary** command.

- The maximum number of BFD sessions supported on any one card is 1250.
- The maximum number of members in a bundle is 64.
- When using BFD with OSPF, consider the following guidelines:
 - BFD establishes sessions from a neighbor to a designated router (DR) or backup DR (BDR) only when the neighbor state is *full*.
 - BFD does not establish sessions between DR-Other neighbors (for example, when their OSPF states are both 2-way).



Caution

If you are using BFD with Unicast Reverse Path Forwarding (uRPF) on a particular interface, then you need to use the **echo disable** command to disable echo mode on that interface; otherwise, echo packets will be rejected. For more information, see the [Disabling Echo Mode](#). To enable or disable IPv4 uRPF checking on an IPv4 interface, use the **[no] ipv4 verify unicast source reachable-via** command in interface configuration mode.

Configuring BFD Under a Dynamic Routing Protocol or Using a Static Route

Enabling BFD on a BGP Neighbor

BFD can be enabled per neighbor, or per interface. This task describes how to enable BFD for BGP on a neighbor router. To enable BFD per interface, use the steps in the [Enabling BFD for OSPF on an Interface](#).



Note

BFD neighbor router configuration is supported for BGP only.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bfd minimum-interval** *milliseconds*
4. **bfd multiplier** *multiplier*
5. **neighbor** *ip-address*
6. **remote-as** *autonomous-system-number*
7. **bfd fast-detect**
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>autonomous-system-number</i> Example: RP/0/RP0/CPU0:router(config)# router bgp 120	Enters BGP configuration mode, allowing you to configure the BGP routing process. Use the show bgp command in EXEC mode to obtain the <i>autonomous-system-number</i> for the current router.
Step 3	bfd minimum-interval <i>milliseconds</i> Example: RP/0/RP0/CPU0:router(config-bgp)# bfd minimum-interval 6500	Sets the BFD minimum interval. Range is 15-30000 milliseconds.
Step 4	bfd multiplier <i>multiplier</i> Example: RP/0/RP0/CPU0:router(config-bgp)# bfd multiplier 7	Sets the BFD multiplier.
Step 5	neighbor <i>ip-address</i> Example: RP/0/RP0/CPU0: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. This example configures the IP address 172.168.40.24 as a BGP peer.
Step 6	remote-as <i>autonomous-system-number</i> Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002	Creates a neighbor and assigns it a remote autonomous system. This example configures the remote autonomous system to be 2002.
Step 7	bfd fast-detect Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# bfd fast-detect	Enables BFD between the local networking devices and the neighbor whose IP address you configured to be a BGP peer in Step 5.

	Command or Action	Purpose
		In the example in Step 5, the IP address 172.168.40.24 was set up as the BGP peer. In this example, BFD is enabled between the local networking devices and the neighbor 172.168.40.24.
Step 8	<code>commit</code>	

Enabling BFD for OSPF on an Interface

The following procedures describe how to configure BFD for Open Shortest Path First (OSPF) on an interface. The steps in the procedure are common to the steps for configuring BFD on IS-IS and MPLS-TE; only the command mode differs.



Note BFD per interface configuration is supported for OSPF, OSPFv3, IS-IS, and MPLS-TE only. For information about configuring BFD on an OSPFv3 interface, see [Enabling BFD for OSPFv3 on an Interface](#).

SUMMARY STEPS

1. `configure`
2. `bfd multipath include locationnode-id`
3. `router ospf process-name`
4. `bfd minimum-interval milliseconds`
5. `bfd multiplier multiplier`
6. `area area-id`
7. `interface type interface-path-id`
8. `bfd fast-detect`
9. `commit`
10. `show run router ospf`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	<p><code>bfd multipath include locationnode-id</code></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config)# bfd multipath include location 0/0/CPU0</pre>	<p>(Optional) Enables BFD multipath for the specified bundle on the interface. This step is required for bundle interfaces.</p> <p>Note</p> <ul style="list-style-type: none"> • This step must be repeated for every line card that has a member link in the bundle interface.
Step 3	<p><code>router ospf process-name</code></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config)# router ospf 0</pre>	<p>Enters OSPF configuration mode, allowing you to configure the OSPF routing process.</p> <p>Use the show ospf command in EXEC configuration mode to obtain the process-name for the current router.</p>

	Command or Action	Purpose
		Note <ul style="list-style-type: none"> To configure BFD for IS-IS or MPLS-TE, enter the corresponding configuration mode. For example, for MPLS-TE, enter MPLS-TE configuration mode.
Step 4	bfd minimum-interval <i>milliseconds</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# bfd minimum-interval 6500</pre>	Sets the BFD minimum interval. Range is 15-30000 milliseconds. This example sets the BFD minimum interval to 6500 milliseconds.
Step 5	bfd multiplier <i>multiplier</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# bfd multiplier 7</pre>	Sets the BFD multiplier. This example sets the BFD multiplier to 7.
Step 6	area <i>area-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# area 0</pre>	Configures an Open Shortest Path First (OSPF) area. Replace <i>area-id</i> with the OSPF area identifier.
Step 7	interface <i>type interface-path-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# interface gigabitEthernet 0/3/0/1</pre>	Enters interface configuration mode and specifies the interface name and notation <i>rack/slot/module/port</i> . <ul style="list-style-type: none"> The example indicates a Gigabit Ethernet interface in modular services card slot 3.
Step 8	bfd fast-detect Example: <pre>RP/0/RP0/CPU0:router(config-ospf-ar-if)# bfd fast-detect</pre>	Enables BFD to detect failures in the path between adjacent forwarding engines.
Step 9	commit	
Step 10	show run router ospf Example: <pre>RP/0/RP0/CPU0:router(config-ospf-ar-if)# show run router ospf</pre>	Verify that BFD is enabled on the appropriate interface.

Enabling BFD for OSPFv3 on an Interface

The following procedures describe how to configure BFD for OSPFv3 on an interface. The steps in the procedure are common to the steps for configuring BFD on IS-IS, and MPLS-TE; only the command mode differs.



Note BFD per-interface configuration is supported for OSPF, OSPFv3, IS-IS, and MPLS-TE only. For information about configuring BFD on an OSPF interface, see [Enabling BFD for OSPF on an Interface](#).

SUMMARY STEPS

1. **configure**
2. **router ospfv3** *process-name*
3. **bfd minimum-interval** *milliseconds*
4. **bfd multiplier** *multiplier*
5. **area** *area-id*
6. **interface** *type interface-path-id*
7. **bfd fast-detect**
8. **commit**
9. **show run router ospfv3**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospfv3 <i>process-name</i> Example: <pre>RP/0/RP0/CPU0:routerconfig)# router ospfv3 0</pre>	<p>Enters OSPFv3 configuration mode, allowing you to configure the OSPFv3 routing process.</p> <p>Use the show ospfv3 command in EXEC mode to obtain the process name for the current router.</p> <p>Note</p> <ul style="list-style-type: none"> • To configure BFD for IS-IS or MPLS-TE, enter the corresponding configuration mode. For example, for MPLS-TE, enter MPLS-TE configuration mode.
Step 3	bfd minimum-interval <i>milliseconds</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospfv3)# bfd minimum-interval 6500</pre>	<p>Sets the BFD minimum interval. Range is 15-30000 milliseconds.</p> <p>This example sets the BFD minimum interval to 6500 milliseconds.</p>
Step 4	bfd multiplier <i>multiplier</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospfv3)# bfd multiplier 7</pre>	<p>Sets the BFD multiplier.</p> <p>This example sets the BFD multiplier to 7.</p>
Step 5	area <i>area-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospfv3)# area 0</pre>	<p>Configures an OSPFv3 area.</p> <p>Replace <i>area-id</i> with the OSPFv3 area identifier.</p>

	Command or Action	Purpose
Step 6	interface <i>type interface-path-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospfv3-ar)# interface gigabitEthernet 0/1/5/0</pre>	Enters interface configuration mode and specifies the interface name and notation <i>rack/slot/module/port</i> . <ul style="list-style-type: none"> The example indicates a Gigabit Ethernet interface in modular services card slot 1.
Step 7	bfd fast-detect Example: <pre>RP/0/RP0/CPU0:router(config-ospfv3-ar-if)# bfd fast-detect</pre>	Enables BFD to detect failures in the path between adjacent forwarding engines.
Step 8	commit	
Step 9	show run router ospfv3 Example: <pre>RP/0/RP0/CPU0:router(config-ospfv3-ar-if)#show run router ospfv3</pre>	Verifies that BFD is enabled on the appropriate interface.

Enabling BFD on a Static Route

The following procedure describes how to enable BFD on a static route.



Note Bundle VLAN sessions are restricted to an interval of 250 milliseconds and a multiplier of 3. More aggressive parameters are not allowed.

SUMMARY STEPS

- configure**
- router static**
- address-family ipv4 unicast *address nexthop* bfd fast-detect [minimum-interval *interval*] [multiplier *multiplier*]**
- vrf *vrf-name***
- address-family ipv4 unicast *address nexthop* bfd fast-detect**
- commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router static Example: <pre>RP/0/RP0/CPU0:router(config)# router static</pre>	Enters static route configuration mode, allowing you to configure static routing.

	Command or Action	Purpose
Step 3	<p>address-family ipv4 unicast <i>address nexthop</i> bfd fast-detect [minimum-interval <i>interval</i>] [multiplier <i>multiplier</i>]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-static)# address-family ipv4 unicast 0.0.0.0/0 2.6.0.1 bfd fast-detect minimum-interval 1000 multiplier 5</pre>	<p>Enables BFD fast-detection on the specified IPV4 unicast destination address prefix and on the forwarding next-hop address.</p> <ul style="list-style-type: none"> • Include the optional minimum-interval keyword and argument to ensure that the next-hop is assigned with the same hello interval. Replace the <i>interval</i> argument with a number that specifies the interval in milliseconds. Range is from 10 through 10000. • Include the optional multiplier keyword argument to ensure that the next hop is assigned with the same detect multiplier. Replace the <i>multiplier</i> argument with a number that specifies the detect multiplier. Range is from 1 through 10. <p>Note Bundle VLAN sessions are restricted to an interval of 250 milliseconds and a multiplier of 3. More aggressive parameters are not allowed.</p>
Step 4	<p>vrf <i>vrf-name</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-static)# vrf vrf1</pre>	<p>Specifies a VPN routing and forwarding (VRF) instance, and enters static route configuration mode for that VRF.</p>
Step 5	<p>address-family ipv4 unicast <i>address nexthop</i> bfd fast-detect</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-static-vrf)# address-family ipv4 unicast 0.0.0.0/0 2.6.0.2</pre>	<p>Enables BFD fast-detection on the specified IPV4 unicast destination address prefix and on the forwarding next-hop address.</p>
Step 6	commit	

Configuring BFD on Bundle Member Links

Prerequisites for Configuring BFD on Bundle Member Links

The physical interfaces that are members of a bundle must be directly connected between peer routers without any switches in between.

Specifying the BFD Destination Address on a Bundle

To specify the BFD destination address on a bundle, complete these steps:

DETAILED STEPS

SUMMARY STEPS

1. **configure**
2. **interface [Bundle-Ether | Bundle-POS] *bundle-id***
3. **bfd address-family ipv4 destination *ip-address***
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	interface [Bundle-Ether Bundle-POS] <i>bundle-id</i> Example: RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1	Enters interface configuration mode for the specified bundle ID.
Step 3	bfd address-family ipv4 destination <i>ip-address</i> Example: RP/0/RP0/CPU0:router(config-if)# bfd address-family ipv4 destination 10.20.20.1	Specifies the primary IPv4 address assigned to the bundle interface on a connected remote system, where <i>ip-address</i> is the 32-bit IP address in dotted-decimal format (A.B.C.D).
Step 4	commit	

Enabling BFD Sessions on Bundle Members

To enable BFD sessions on bundle member links, complete these steps:

SUMMARY STEPS

1. **configure**
2. **interface [Bundle-Ether | Bundle-POS] *bundle-id***
3. **bfd address-family ipv4 fast-detect**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	interface [Bundle-Ether Bundle-POS] <i>bundle-id</i> Example: RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1	Enters interface configuration mode for the specified bundle ID.
Step 3	bfd address-family ipv4 fast-detect Example:	Enables IPv4 BFD sessions on bundle member links.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-if)# bfd address-family ipv4 fast-detect	
Step 4	commit	

Configuring the Minimum Thresholds for Maintaining an Active Bundle

The bundle manager uses two configurable minimum thresholds to determine whether a bundle can be brought up or remain up, or is down, based on the state of its member links.

- Minimum active number of links
- Minimum active bandwidth available

Whenever the state of a member changes, the bundle manager determines whether the number of active members or available bandwidth is less than the minimum. If so, then the bundle is placed, or remains, in DOWN state. Once the number of active links or available bandwidth reaches one of the minimum thresholds, then the bundle returns to the UP state.

To configure minimum bundle thresholds, complete these steps:

SUMMARY STEPS

1. **configure**
2. **interface [Bundle-Ether | Bundle-POS] *bundle-id***
3. **bundle minimum-active bandwidth *kbps***
4. **bundle minimum-active links *links***
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	interface [Bundle-Ether Bundle-POS] <i>bundle-id</i> Example: RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1	Enters interface configuration mode for the specified bundle ID.
Step 3	bundle minimum-active bandwidth <i>kbps</i> Example: RP/0/RP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000	Sets the minimum amount of bandwidth required before a bundle can be brought up or remain up. The range is from 1 through a number that varies depending on the platform and the bundle type.
Step 4	bundle minimum-active links <i>links</i> Example:	Sets the number of active links required before a bundle can be brought up or remain up. The range is from 1 to 32.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2	Note <ul style="list-style-type: none"> When BFD is started on a bundle that is already active, the BFD state of the bundle is declared when the BFD state of all the existing active members is known.
Step 5	commit	

Configuring BFD Packet Transmission Intervals and Failure Detection Times on a Bundle

BFD asynchronous packet intervals and failure detection times for BFD sessions on bundle member links are configured using a combination of the **bfd address-family ipv4 minimum-interval** and **bfd address-family ipv4 multiplier** interface configuration commands on a bundle.

The BFD control packet interval is configured directly using the **bfd address-family ipv4 minimum-interval** command. The BFD echo packet interval and all failure detection times are determined by a combination of the interval and multiplier values in these commands. For more information see the [BFD Packet Intervals and Failure Detection](#).

To configure the minimum transmission interval and failure detection times for BFD asynchronous mode control and echo packets on bundle member links, complete these steps:

DETAILED STEPS

SUMMARY STEPS

1. **configure**
2. **interface [Bundle-Ether | Bundle-POS] bundle-id**
3. **bfd address-family ipv4 minimum-interval milliseconds**
4. **bfd address-family ipv4 multiplier multiplier**
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	interface [Bundle-Ether Bundle-POS] bundle-id Example: RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1	Enters interface configuration mode for the specified bundle ID.
Step 3	bfd address-family ipv4 minimum-interval milliseconds Example: RP/0/RP0/CPU0:router(config-if)#bfd address-family ipv4 minimum-interval 2000	

	Command or Action	Purpose
	<p>Note</p> <ul style="list-style-type: none"> Specifies the minimum interval, in milliseconds, for asynchronous mode control packets on IPv4 BFD sessions on bundle member links. The range is from 15 to 30000. Although the command allows you to configure a minimum of 15 ms, the supported minimum on the Cisco CRS-1 Router is 33 ms. 	
Step 4	<p>bfd address-family ipv4 multiplier <i>multiplier</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-if)#bfd address-family ipv4 multiplier 30</pre>	<p>Specifies a number that is used as a multiplier with the minimum interval to determine BFD control and echo packet failure detection times and echo packet transmission intervals for IPv4 BFD sessions on bundle member links. The range is from 2 to 50. The default is 3.</p> <p>Note</p> <ul style="list-style-type: none"> Although the command allows you to configure a minimum of 2, the supported minimum is 3.
Step 5	commit	

Configuring Allowable Delays for BFD State Change Notifications Using Timers on a Bundle

The BFD system supports two configurable timers to allow for delays in receipt of BFD SCNs from peers before declaring a BFD session on a link bundle member down:

- BFD session startup
- BFD configuration removal by a neighbor

For more information about how these timers work and other BFD state change behavior, see the [Overview of BFD State Change Behavior on Member Links and Bundle Status](#).

To configure the timers that allow for delays in receipt of BFD SCNs from peers, complete these steps:

SUMMARY STEPS

1. **configure**
2. **interface** [**Bundle-Ether** | **Bundle-POS**] *bundle-id*
3. **bfd address-family ipv4 timers start** *seconds*
4. **bfd address-family ipv4 timers nbr-unconfig** *seconds*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	

	Command or Action	Purpose
Step 2	interface [Bundle-Ether Bundle-POS] <i>bundle-id</i> Example: <pre>RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1</pre>	Enters interface configuration mode for the specified bundle ID.
Step 3	bfd address-family ipv4 timers start <i>seconds</i> Example: <pre>RP/0/RP0/CPU0:router(config-if)#</pre>	Specifies the number of seconds after startup of a BFD member link session to wait for the expected notification from the BFD peer to be received, so that the session can be declared up. If the SCN is not received after that period of time, the BFD session is declared down. The range is 60 to 3600. (In Cisco IOS XR Releases 4.0 and 4.0.1, the available minimum is 30, but is not recommended.)
Step 4	bfd address-family ipv4 timers nbr-unconfig <i>seconds</i> Example: <pre>RP/0/RP0/CPU0:router(config-if)#</pre>	Specifies the number of seconds to wait after receipt of notification that BFD configuration has been removed by a BFD neighbor, so that any configuration inconsistency between the BFD peers can be fixed. If the BFD configuration issue is not resolved before the specified timer is reached, the BFD session is declared down. The range is 30 to 3600.
Step 5	commit	

Configure BFD over Bundles CISCO/IETF Mode Support on a Per Bundle Basis

To configure BFD over Bundles CISCO/IETF mode support on a per bundle basis use these steps:

Before you begin

The BFD mode change (Cisco to IETF and vice-versa) goes through when a bundle is newly created or only when the BFD state for the bundle is 'down' or 'BoB nonoperational.'



Note This procedure is applicable from release 5.3.1 onwards.

SUMMARY STEPS

1. **configure**
2. **interface [Bundle-Ether | Bundle-POS] *bundle-id***
3. **no bfd address-family ipv4 fast-detect**
4. **commit**
5. **bfd mode { cisco | ietf }**
6. **bfd address-family ipv4 fast-detect**
7. **commit**
8. **show bundle bundle-ether *bundle-id***

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	interface [Bundle-Ether Bundle-POS] <i>bundle-id</i> Example: RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1	Enters interface configuration mode for the specified bundle ID.
Step 3	no bfd address-family ipv4 fast-detect Example: RP/0/RP0/CPU0:router(config-if)# no bfd address-family ipv4 fast-detect	Disables IPv4 BFD sessions on the specified bundle.
Step 4	commit	
Step 5	bfd mode { cisco ietf } Example: RP/0/RP0/CPU0:router(config-if)# bfd mode ietf	Enables Cisco or IETF mode for BFD over bundle for the specified bundle. Default is cisco .
Step 6	bfd address-family ipv4 fast-detect Example: RP/0/RP0/CPU0:router(config-if)# bfd address-family ipv4 fast-detect	Enables IPv4 BFD sessions on the specified bundle.
Step 7	commit	
Step 8	show bundle bundle-ether <i>bundle-id</i>	Displays the selected bundle mode.

Sample show command output to check the mode

This example show the output of the **show bundle bundle-ether** command with the bundle mode selected:

```
RP/0/RP0/CPU0:R3-PE3#sh bundle bundle-ether 4301

Bundle-Ether4301
  Status:                               Up
  Local links {active/standby/configured}: 2 / 0 / 2
  Local bandwidth {effective/available}:    20000000 (20000000) kbps
  MAC address (source):                    0014.1c00.0003 (Chassis pool)
  Inter-chassis link:                       No
  Minimum active links / bandwidth:         1 / 1 kbps
  Maximum active links:                     64
  Wait while timer:                          2000 ms
  Load balancing:                            Default
  LACP:                                       Operational
  Flap suppression timer:                   Off
```

```

Cisco extensions:                Disabled
mLACP:                          Not configured
IPv4 BFD:                       Operational
State:                          Up
Mode:                            ietf #####----- this is the mode
cisco/ietf .
Fast detect:                    Enabled
Start timer:                   60 s
Neighbor-unconfigured timer:   60 s
Preferred min interval:       150 ms
Preferred multiple:            3
Destination address:          101.43.1.1

Port          Device          State          Port ID          B/W, kbps
-----
Te0/5/0/4    Local          Active         0x8000, 0x0012  10000000
  Link is Active
Te0/7/0/8    Local          Active         0x8000, 0x0006  10000000
  Link is Active

```

What to do next

For a bundle to accept the new BFD mode change, you must bring down and then recreate the existing BFD sessions.

Enabling Echo Mode to Test the Forwarding Path to a BFD Peer

BFD echo mode is enabled by default for the following interfaces:

- For IPv4 on member links of BFD bundle interfaces.
- For IPv4 on other physical interfaces whose minimum interval is less than two seconds.



Note If you have configured a BFD minimum interval greater than two seconds on a physical interface using the **bfd minimum-interval** command, then you will need to change the interval to be less than two seconds to support and enable echo mode. This does not apply to bundle member links, which always support echo mode.

Overriding the Default Echo Packet Source Address

If you do not specify an echo packet source address, then BFD uses the IP address of the output interface as the default source address for an echo packet.

In Cisco IOS XR releases before 3.9.0, we recommend that you configure the local router ID using the **router-id** command to change the default IP address for the echo packet source address to the address specified as the router ID.

Beginning in Cisco IOS XR release 3.9.0 and later, you can use the **echo ipv4 source** command in BFD or interface BFD configuration mode to specify the IP address that you want to use as the echo packet source address.

You can override the default IP source address for echo packets for BFD on the entire router, or for a particular interface.

Specifying the Echo Packet Source Address Globally for BFD

To specify the echo packet source IP address globally for BFD on the router, complete the following steps:

SUMMARY STEPS

1. **configure**
2. **bfd**
3. **echo ipv4 source *ip-address***
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	bfd Example: RP/0/RP0/CPU0:router(config)# bfd	Enters BFD configuration mode.
Step 3	echo ipv4 source <i>ip-address</i> Example: RP/0/RP0/CPU0:router(config-bfd)# echo ipv4 source 10.10.10.1	Specifies an IPv4 address to be used as the source address in BFD echo packets, where <i>ip-address</i> is the 32-bit IP address in dotted-decimal format (A.B.C.D).
Step 4	commit	

Specifying the Echo Packet Source Address on an Individual Interface or Bundle

To specify the echo packet source IP address on an individual BFD interface or bundle, complete the following steps:

SUMMARY STEPS

1. **configure**
2. **bfd**
3. **interface type interface-path-id**
4. **echo ipv4 source *ip-address***
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	

	Command or Action	Purpose
Step 2	bfd Example: RP/0/RP0/CPU0:router(config)# bfd	Enters BFD configuration mode.
Step 3	interface type interface-path-id Example: RP/0/RP0/CPU0:router(config-bfd)# interface gigabitEthernet 0/1/5/0	Enters BFD interface configuration mode for a specific interface or bundle. In BFD interface configuration mode, you can specify an IPv4 address on an individual interface or bundle.
Step 4	echo ipv4 source ip-address Example: RP/0/RP0/CPU0:router(config-bfd)# echo ipv4 source 10.10.10.1	Specifies an IPv4 address to be used as the source address in BFD echo packets, where <i>ip-address</i> is the 32-bit IP address in dotted-decimal format (A.B.C.D).
Step 5	commit	

Configuring BFD Session Teardown Based on Echo Latency Detection

Beginning in Cisco IOS XR 4.0.1, you can configure BFD sessions on non-bundle interfaces to bring down a BFD session when it exceeds the configured echo latency tolerance.

To configure BFD session teardown using echo latency detection, complete the following steps.

Before you enable echo latency detection, be sure that your BFD configuration supports echo mode.

Echo latency detection is not supported on bundle interfaces.

DETAILED STEPS

SUMMARY STEPS

1. **configure**
2. **bfd**
3. **echo latency detect** [percentage *percent-value* [count *packet-count*]]
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	bfd Example: RP/0/RP0/CPU0:router(config)# bfd	Enters BFD configuration mode.

	Command or Action	Purpose
Step 3	echo latency detect [<i>percentage percent-value</i> [<i>count packet-count</i>]] Example: <pre>RP/0/RP0/CPU0:router(config-bfd)# echo latency detect</pre>	Enables echo packet latency detection over the course of a BFD session, where: <ul style="list-style-type: none"> • percentage <i>percent-value</i>—Specifies the percentage of the echo failure detection time to be detected as bad latency. The range is 100 to 250. The default is 100. • count <i>packet-count</i>—Specifies a number of consecutive packets received with bad latency that will take down a BFD session. The range is 1 to 10. The default is 1.
Step 4	commit	

Delaying BFD Session Startup Until Verification of Echo Path and Latency

Beginning in Cisco IOS XR Release 4.0.1, you can verify that the echo packet path is working and within configured latency thresholds before starting a BFD session on non-bundle interfaces.



Note Echo startup validation is not supported on bundle interfaces.

To configure BFD echo startup validation, complete the following steps.

Before you begin

Before you enable echo startup validation, be sure that your BFD configuration supports echo mode.

SUMMARY STEPS

1. **configure**
2. **bfd**
3. **echo startup validate** [*force*]
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	bfd Example: <pre>RP/0/0RP0RSP0/CPU0:router(config)# bfd</pre>	Enters BFD configuration mode.
Step 3	echo startup validate [<i>force</i>] Example:	Enables verification of the echo packet path before starting a BFD session, where an echo packet is periodically transmitted on the link to verify successful transmission

	Command or Action	Purpose
	RP/0/0RP0RSP0/CPU0:router(config-bfd)# echo startup validate	<p>within the configured latency before allowing the BFD session to change state.</p> <p>When the force keyword is not configured, the local system performs echo startup validation if the following conditions are true:</p> <ul style="list-style-type: none"> • The local router is capable of running echo (echo is enabled for this session). • The remote router is capable of running echo (received control packet from remote system has non-zero "Required Min Echo RX Interval" value). <p>When the force keyword is configured, the local system performs echo startup validation if following conditions are true.</p> <ul style="list-style-type: none"> • The local router is capable of running echo (echo is enabled for this session). • The remote router echo capability is not considered (received control packet from remote system has zero or non-zero "Required Min Echo RX Interval" value).
Step 4	commit	

Disabling Echo Mode

BFD does not support asynchronous operation in echo mode in certain environments. Echo mode should be disabled when using BFD for the following applications or conditions:

- BFD with uRPF (IPv4)
- To support rack reload and online insertion and removal (OIR) when a BFD bundle interface has member links that span multiple racks.



Note BFD echo mode is automatically disabled for BFD on physical interfaces when the minimum interval is greater than two seconds. The minimum interval does not affect echo mode on BFD bundle member links. BFD echo mode is also automatically disabled for BFD on bundled VLANs and IPv6 (global and link-local addressing).

You can disable echo mode for BFD on the entire router, or for a particular interface.

Disabling Echo Mode on a Router

To disable echo mode globally on the router complete the following steps:

DETAILED STEPS

SUMMARY STEPS

1. **configure**
2. **bfd**
3. **echo disable**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	bfd Example: RP/0/RP0/CPU0:router(config)# bfd	Enters BFD configuration mode.
Step 3	echo disable Example: RP/0/RP0/CPU0:router(config-bfd)# echo disable	Disables echo mode on the router.
Step 4	commit	

Disabling Echo Mode on an Individual Interface or Bundle

The following procedures describe how to disable echo mode on an interface or bundle .

SUMMARY STEPS

1. **configure**
2. **bfd**
3. **interface** *type interface-path-id*
4. **echo disable**
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	bfd Example: RP/0/RP0/CPU0:router(config)# bfd	Enters BFD configuration mode.
Step 3	interface <i>type interface-path-id</i> Example:	Enters BFD interface configuration mode for a specific interface or bundle. In BFD interface configuration mode, you can disable echo mode on an individual interface or bundle.

	Command or Action	Purpose
	<pre>RP/0/RP0/CPU0:router(config-bfd)# interface gigabitEthernet 0/1/5/0</pre>	
Step 4	echo disable Example: <pre>RP/0/RP0/CPU0:router(config-bfd-if)# echo disable</pre>	Disables echo mode on the specified individual interface or bundle.
Step 5	commit	

Minimizing BFD Session Flapping Using BFD Dampening

To configure BFD dampening to control BFD session flapping, complete the following steps.

SUMMARY STEPS

1. **configure**
2. **bfd**
3. **dampening [bundle-member] {initial-wait | maximum-wait | secondary-wait} milliseconds**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	bfd Example: <pre>RP/0/RP0/CPU0:router(config)# bfd</pre>	Enters BFD configuration mode.
Step 3	dampening [bundle-member] {initial-wait maximum-wait secondary-wait} milliseconds Example: <pre>RP/0/RP0/CPU0:router(config-bfd)# dampening initial-wait 30000</pre>	<p>Specifies delays in milliseconds for BFD session startup to control flapping.</p> <p>The value for maximum-wait should be greater than the value for initial-wait.</p> <p>The dampening values can be defined for bundle member interfaces and for the non-bundle interfaces.</p>
Step 4	commit	

Enabling and Disabling IPv6 Checksum Support

By default, IPv6 checksum calculations on UDP packets are enabled for BFD on the router.

You can disable IPv6 checksum support for BFD either on the entire router, or for a particular interface. A misconfiguration may occur if the IPv6 checksum support is enabled at one router, but disabled at the other. Therefore, you should enable or disable IPv6 checksum support at both the routers.

These sections describe about:



Note The command-line interface (CLI) is slightly different in BFD configuration and BFD interface configuration. For BFD configuration, the **disable** keyword is not optional. Therefore, to enable BFD configuration in that mode, you need to use the **no** form of the command.

Enabling and Disabling IPv6 Checksum Calculations for BFD on a Router

To enable or disable IPv6 checksum calculations globally on the router complete the following steps:

SUMMARY STEPS

1. **configure**
2. **bfd**
3. **ipv6 checksum [disable]**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	bfd Example: RP/0/RP0/CPU0:router (config) # bfd	Enters BFD configuration mode.
Step 3	ipv6 checksum [disable] Example: RP/0/RP0/CPU0:router (config-bfd-if) # ipv6 checksum disable	Enables IPv6 checksum support on the interface. To disable, use the disable keyword.
Step 4	commit	

Enabling and Disabling IPv6 Checksum Calculations for BFD on an Individual Interface or Bundle

The following procedures describe how to enable or disable IPv6 checksum calculations on an interface or bundle .

DETAILED STEPS

SUMMARY STEPS

1. **configure**
2. **bfd**
3. **interface type interface-path-id**
4. **ipv6 checksum [disable]**

5. commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	bfd Example: RP/0/RP0/CPU0:router(config)# bfd	Enters BFD configuration mode.
Step 3	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-bfd)# interface gigabitEthernet 0/1/5/0	Enters BFD interface configuration mode for a specific interface or bundle.
Step 4	ipv6 checksum [disable] Example: RP/0/RP0/CPU0:router(config-bfd-if)# ipv6 checksum	Enables IPv6 checksum support on the interface. To disable, use the disable keyword.
Step 5	commit	

Clearing and Displaying BFD Counters

The following procedure describes how to display and clear BFD packet counters. You can clear packet counters for BFD sessions that are hosted on a specific node or on a specific interface.

SUMMARY STEPS

1. **show bfd counters** [**ipv4** | **ipv6** | **all**] **packet interface** *type interface-path-id* **location** *node-id*
2. **clear bfd counters** [**ipv4** | **ipv6** | **all**] **packet** [**interface** *type interface-path-id*] **location** *node-id*
3. **show bfd counters** [**ipv4** | **ipv6** | **all**] **packet** [**interface** *type interface-path-id*] **location** *node-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	show bfd counters [ipv4 ipv6 all] packet interface <i>type interface-path-id</i> location <i>node-id</i> Example: RP/0/RP0/CPU0:router#show bfd counters all packet location 0/3/cpu0	Displays the BFD counters for IPv4 packets, IPv6 packets, or all packets.

	Command or Action	Purpose
Step 2	clear bfd counters [ipv4 ipv6 all] packet [interface type interface-path-id] location node-id Example: RP/0/RP0/CPU0:router# clear bfd counters all packet location 0/3/cpu0	Clears the BFD counters for IPv4 packets, IPv6 packets, or all packets.
Step 3	show bfd counters [ipv4 ipv6 all] packet [interface type interface-path-id] location node-id Example: RP/0/RP0/CPU0:router# show bfd counters all packet location 0/3/cpu0	Verifies that the BFD counters for IPv4 packets, IPv6 packets, or all packets have been cleared.

Configuring Coexistence Between BFD over Bundle (BoB) and BFD over Logical Bundle (BLB)

Perform this task to configure the coexistence mechanism between BoB and BLB:

Before you begin

You must configure one or more linecards to allow hosting of MP BFD sessions. If no linecards are included, linecards groups will not be formed, and consequently no BFD MP sessions are created. For default settings of group size and number, at least two lines with the **bfd multipath include location node-id** command and valid line cards must be added to the configuration for the algorithm to start forming groups and BFD MP sessions to be established.

As sample configuration is provided:

```
(config)#bfd multipath include location 0/0/CPU0
(config)#bfd multipath include location 0/1/CPU0
```

SUMMARY STEPS

1. **configure**
2. **bfd**
3. Use one of these commands:
 - **bundle coexistence bob-blb inherit**
 - **bundle coexistence bob-blb logical**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	

	Command or Action	Purpose
Step 2	bfd Example: RP/0/RP0/CPU0:router(config)#bfd	Configures Bi-directional Forwarding Detection (BFD) and enters global BFD configuration mode.
Step 3	Use one of these commands: <ul style="list-style-type: none"> • bundle coexistence bob-blb inherit • bundle coexistence bob-blb logical Example: RP/0/RP0/CPU0:router(config-bfd)#bundle coexistence bob-blb inherit Or RP/0/RP0/CPU0:router(config-bfd)#bundle coexistence bob-blb logical	Configures the coexistence mechanism between BoB and BLB. <ul style="list-style-type: none"> • inherit—Use the inherit keyword to configure "inherited" coexistence mode. • logical—Use the logical keyword to configure "logical" coexistence mode.
Step 4	commit	

Configuring BFD over MPLS Traffic Engineering LSPs

Enabling BFD Parameters for BFD over TE Tunnels

BFD for TE tunnel is enabled at the head-end by configuring BFD parameters under the tunnel. When BFD is enabled on the already up tunnel, TE waits for the bringup timeout before bringing down the tunnel. BFD is disabled on TE tunnels by default. Perform these tasks to configure BFD parameters and enable BFD over TE Tunnels.



Note BFD paces the creation of BFD sessions by limiting LSP ping messages to be under 50 PPS to avoid variations in CPU usage.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *interface-number*
3. **bfd fast-detect**
4. **bfd minimum-interval***milliseconds*
5. **bfd multiplier** *number*
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	

	Command or Action	Purpose
Step 2	interface tunnel-te <i>interface-number</i> Example: RP/0/RP0/CPU0:router(config)#interface tunnel-te 65535	Configures MPLS Traffic Engineering (MPLS TE) tunnel interface and enters into MPLS TE tunnel interface configuration mode.
Step 3	bfd fast-detect Example: RP/0/RP0/CPU0:router(config-if)#bfd fast-detect	Enables BFD fast detection.
Step 4	bfd minimum-interval <i>milliseconds</i> Example: RP/0/RP0/CPU0:router(config-if)#bfd minimum-interval 2000	Configures hello interval in milliseconds. Hello interval range is 100 to 30000 milliseconds. Default hello interval is 100 milliseconds
Step 5	bfd multiplier <i>number</i> Example: RP/0/RP0/CPU0:router(config-if)#bfd multiplier 5	Configures BFD multiplier detection. BFD multiplier range is 3 to 10. Default BFD multiplier is 3.
Step 6	commit	

What to do next

Configure BFD bring up timeout interval.

Once LSP is signaled and BFD session is created, TE allows given time for the BFD session to come up. If BFD session fails to come up within timeout, the LSP is torn down. Hence it is required to configure BFD bring up timeout

Configuring BFD Bring up Timeout

Perform these steps to configure BFD bring up timeout interval. The default bring up timeout interval is 60 seconds.

Before you begin

BFD must be enabled under MPLS TE tunnel interface.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *interface-number*
3. **bfd bringup-timeout** *seconds*
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	

	Command or Action	Purpose
Step 2	interface tunnel-te <i>interface-number</i> Example: RP/0/RP0/CPU0:router(config)#interface tunnel-te 65535	Configures MPLS Traffic Engineering (MPLS TE) tunnel interface and enters into MPLS TE tunnel interface configuration mode.
Step 3	bfd bringup-timeout <i>seconds</i> Example: RP/0/RP0/CPU0:router(config-if)#bfd bringup-timeout 2400	Enables the time interval (in seconds) to wait for the BFD session to come up. Bring up timeout range is 6 to 3600 seconds. Default bring up timeout interval is 60 seconds.
Step 4	commit	

What to do next

Configure BFD dampening parameters to bring up the TE tunnel and to avoid signaling churn in the network.

Configuring BFD Dampening for TE Tunnels

When BFD session fails to come up, TE exponentially backs off using the failed path-option to avoid signaling churn in the network.

Perform these steps to configure dampening intervals to bring the TE tunnel up.

Before you begin

- BFD must be enabled under MPLS TE tunnel interface.
- BFD bring up timeout interval must be configured using the **bfd bringup-timeout** command.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *interface-number*
3. **bfd dampening initial-wait** *milliseconds*
4. **bfd dampening maximum-wait** *milliseconds*
5. **bfd dampening secondary-wait** *milliseconds*
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	interface tunnel-te <i>interface-number</i> Example: RP/0/RP0/CPU0:router(config)#interface tunnel-te 65535	Configures MPLS Traffic Engineering (MPLS TE) tunnel interface and enters into MPLS TE tunnel interface configuration mode.

	Command or Action	Purpose
Step 3	bfd dampening initial-wait <i>milliseconds</i> Example: RP/0/RP0/CPU0:router(config-if)#bfd dampening initial-wait 360000	Configures the initial delay interval before bringing up the tunnel. The initial-wait bring up delay time interval range is 1 to 518400000 milliseconds. Default initial-wait interval is 16000 milliseconds. Note This option brings up the TE tunnel with the previous signaled bandwidth.
Step 4	bfd dampening maximum-wait <i>milliseconds</i> Example: RP/0/RP0/CPU0:router(config-if)#bfd dampening maximum-wait 700000	Configures the maximum delay interval before bringing up the tunnel. The maximum-wait bring up delay time interval range is 1 to 518400000 milliseconds. Default initial-wait interval is 600000 milliseconds. Note This option brings up the TE tunnel with the configured bandwidth.
Step 5	bfd dampening secondary-wait <i>milliseconds</i> Example: RP/0/RP0/CPU0:router(config-if)#bfd dampening secondary-wait 30000	Configures the secondary delay interval before bringing up the tunnel. The secondary-wait bring up delay time interval range is 1 to 518400000 milliseconds. Default secondary-wait interval is 20000 milliseconds.
Step 6	commit	

What to do next

Configure periodic LSP ping option.

Configuring Periodic LSP Ping Requests

Perform this task to configure sending periodic LSP ping requests with BFD TLV, after BFD session comes up.

Before you begin

BFD must be enabled under MPLS TE tunnel interface.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *interface-number*
3. Use one of these commands:
 - **bfd lsp-ping interval** *300*
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	interface tunnel-te <i>interface-number</i> Example: <pre>RP/0/RP0/CPU0:router(config)#interface tunnel-te 65535</pre>	Configures MPLS Traffic Engineering (MPLS TE) tunnel interface and enters into MPLS TE tunnel interface configuration mode.
Step 3	Use one of these commands: <ul style="list-style-type: none"> • bfd lsp-ping interval <i>300</i> Example: <pre>RP/0/RP0/CPU0:router(config-if)#bfd lsp-ping interval 300</pre> Or <pre>RP/0/RP0/CPU0:router(config-if)#bfd lsp-ping disable</pre>	Sets periodic interval for LSP ping requests or disables LSP ping requests. <ul style="list-style-type: none"> • interval <i>seconds</i>—Sets periodic LSP ping request interval in seconds. The interval range is 60 to 3600 seconds. Default interval is 120 seconds. • disable—Disables periodic LSP ping requests. Periodic LSP ping request is enabled by default. The default interval for ping requests is 120 seconds. BFD paces LSP ping to be under 50 ping per seconds (PPS). Thus ping interval is honored; however, this is not guaranteed unless configuring an interval between 60 and 3600 seconds.
Step 4	<code>commit</code>	

What to do next

Configure BFD at the tail-end.

Configuring BFD at the Tail End

Use the tail end global configuration commands to set the BFD minimum-interval and BFD multiplier parameters for all BFD over LSP sessions. The ranges and default values are the same as the BFD head end configuration values. BFD will take the maximum value set between head end minimum interval and tail end minimum interval.

Perform these tasks to configure BFD at the tail end.

SUMMARY STEPS

1. `configure`
2. `mpls traffic-eng bfd lsp tailminimum-interval` *milliseconds*
3. `mpls traffic-eng bfd lsp tailmultiplier` *number*
4. `commit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	

	Command or Action	Purpose
Step 2	mpls traffic-eng bfd lsp tailminimum-interval <i>milliseconds</i> Example: RP/0/RP0/CPU0:router(config)#mpls traffic-eng bfd lsp tail minimum-interval 20000	Configures hello interval in milliseconds. Hello interval range is 100 to 30000 milliseconds. Default hello interval is 100 milliseconds
Step 3	mpls traffic-eng bfd lsp tailmultiplier <i>number</i> Example: RP/0/RP0/CPU0:router(config)#mpls traffic-eng bfd lsp tail multiplier 5	Configures BFD multiplier detection. BFD multiplier detect range is 3 to 10. Default BFD multiplier is 3.
Step 4	commit	

What to do next

Configure **bfd multipath include location** *node-id* command to include specified line cards to host BFD multiple path sessions.

Configuring BFD over LSP Sessions on Line Cards

BFD over LSP sessions, both head-end and tail-end, will be hosted on line cards with following configuration enabled.

SUMMARY STEPS

1. **configure**
2. **bfd**
3. **multipath include location** *node-id*
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	bfd Example: RP/0/RP0/CPU0:router(config)# bfd	Enters BFD configuration mode.
Step 3	multipath include location <i>node-id</i> Example: RP/0/RP0/CPU0:router(config-bfd)# multipath include location 0/1/CPU0	Configures BFD multiple path on specific line card. One or more line cards must be configured with bfd multipath include . For example, <pre>bfd multipath include location 0/1/CPU0 multipath include location 0/2/CPU0</pre> BFD over LSP sessions, both head-end and tail-end, will be hosted on line cards. BFD over LSP sessions, both

	Command or Action	Purpose
		head-end and tail-end, will be distributed to line cards 0/1/CPU0 and 0/2/CPU0 according to internal selection mechanism.
Step 4	commit	

Configuring BFD Object Tracking:

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type bfdtr rate** *tx-rate*
4. **debouncedebounce**
5. **interface** *if-name*
6. **destaddress** *dest_addr*
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	track <i>track-name</i> Example: RP/0/RP0/CPU0:router(config)# track track1	Enters track configuration mode. • <i>track-name</i> —Specifies a name for the object to be tracked.
Step 3	type bfdtr rate <i>tx-rate</i> Example: RP/0/RP0/CPU0:router(config-track)# type bfdtr rate 4	<i>tx_rate</i> - time in msec at which the BFD should probe the remote entity
Step 4	debouncedebounce Example: RP/0/RP0/CPU0:router(config-if)# debounce 10	<i>debounce</i> - count of consecutive BFD probes whose status should match before BFD notifies OT
Step 5	interface <i>if-name</i> Example: RP/0/RP0/CPU0:router(config-track-line-prot)# interface atm 0/2/0/0.1	<i>if_name</i> - interface name on the source to be used by BFD to check the remote BFD status.
Step 6	destaddress <i>dest_addr</i> Example:	<i>dest_addr</i> - IPV4 address of the remote BFD entity being tracked.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-if)#destaddress 1.2.3.4	
Step 7	commit	

Configuration Examples for Configuring BFD

BFD Over BGP: Example

The following example shows how to configure BFD between autonomous system 65000 and neighbor 192.168.70.24:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#router bgp 65000
RP/0/RP0/CPU0:router(config-bgp)#bfd multiplier 2
RP/0/RP0/CPU0:router(config-bgp)#bfd minimum-interval 20
RP/0/RP0/CPU0:router(config-bgp)#neighbor 192.168.70.24
RP/0/RP0/CPU0:router(config-bgp-nbr)#remote-as 2
RP/0/RP0/CPU0:router(config-bgp-nbr)#bfd fast-detect
RP/0/RP0/CPU0:router(config-bgp-nbr)#commit
RP/0/RP0/CPU0:router(config-bgp-nbr)#end
RP/0/RP0/CPU0:router#show run router bgp
```

BFD Over OSPF: Examples

The following example shows how to enable BFD for OSPF on a Gigabit Ethernet interface:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#router ospf 0
RP/0/RP0/CPU0:router(config-ospf)#area 0
RP/0/RP0/CPU0:router(config-ospf-ar)#interface GigabitEthernet 0/3/0/1
RP/0/RP0/CPU0:router(config-ospf-ar-if)#bfd fast-detect
RP/0/RP0/CPU0:router(config-ospf-ar-if)#commit
RP/0/RP0/CPU0:router(config-ospf-ar-if)#end

RP/0/RP0/CPU0:router#show run router ospf

router ospf 0
area 0
interface GigabitEthernet0/3/0/1
bfd fast-detect
```

The following example shows how to enable BFD for OSPFv3 on a Gigabit Ethernet interface:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#router ospfv3 0
RP/0/RP0/CPU0:router(config-ospfv3)#bfd minimum-interval 6500
RP/0/RP0/CPU0:router(config-ospfv3)#bfd multiplier 7
RP/0/RP0/CPU0:router(config-ospfv3-ar)#area 0
RP/0/RP0/CPU0:router(config-ospfv3-ar)#interface gigabitethernet 0/1/5/0
```

```
RP/0/RP0/CPU0:router(config-ospfv3-ar-if)#bfd fast-detect
RP/0/RP0/CPU0:router(config-ospfv3-ar-if)#commit
RP/0/RP0/CPU0:router(config-ospfv3-ar-if)#end

RP/0/RP0/CPU0:router#show run router ospfv3
router ospfv3
area 0
interface GigabitEthernet0/1/5/0
bfd fast-detect
```

BFD Over Static Routes: Examples

The following example shows how to enable BFD on an IPv4 static route. In this example, BFD sessions are established with the next-hop 10.3.3.3 when it becomes reachable.

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#router static
RP/0/RP0/CPU0:router(config-static)#address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-static)#10.2.2.0/24 10.3.3.3 bfd fast-detect
RP/0/RP0/CPU0:router(config-static)#end
```

The following example shows how to enable BFD on an IPv6 static route. In this example, BFD sessions are established with the next hop 2001:0DB8:D987:398:AE3:B39:333:783 when it becomes reachable.

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#router static
RP/0/RP0/CPU0:router(config-static)#address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-static)#2001:0DB8:C18:2:1::F/64 2001:0DB8:D987:398:AE3:B39:333:783
  bfd fast-detect minimum-interval 150 multiplier 4
RP/0/RP0/CPU0:router(config-static)#end

RP/0/RP0/CPU0:router#show run router static address-family ipv6 unicast
```

BFD on Bundled VLANs: Example

The following example shows how to configure BFD on bundled VLANs:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#bfd
RP/0/RP0/CPU0:router(config-bfd)#multipath include location 0/0/CPU0
RP/0/RP0/CPU0:router(config-bfd)#exit

RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#interface Bundle-ether 1
RP/0/RP0/CPU0:router(config-if)#bundle maximum-active links 1
RP/0/RP0/CPU0:router(config-if)#exit
!
RP/0/RP0/CPU0:router(config)#interface TenGigE 0/1/0/1
RP/0/RP0/CPU0:router(config-if)#bundle id 1 mode active
RP/0/RP0/CPU0:router(config-if)#exit
!
RP/0/RP0/CPU0:router(config)#interface TenGigE 0/2/0/1
RP/0/RP0/CPU0:router(config-if)#bundle id 1 mode active
RP/0/RP0/CPU0:router(config-if)#exit
```

```

!
RP/0/RP0/CPU0:router(config)#interface Bundle-Ether1.2
RP/0/RP0/CPU0:router(config-if)#ipv4 address 172.16.2.1 255.255.255.0
RP/0/RP0/CPU0:router(config-if)#encapsulation dot1q 2
RP/0/RP0/CPU0:router(config-if)#exit
!
RP/0/RP0/CPU0:router(config)#interface Bundle-Ether1.1
RP/0/RP0/CPU0:router(config-if)#ipv4 address 172.16.1.1 255.255.255.0
RP/0/RP0/CPU0:router(config-if)#encapsulation dot1q 1
!
RP/0/RP0/CPU0:router(config)#router static
RP/0/RP0/CPU0:router(config-static)#address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-static-afi)#10.2.1.0/24 172.16.1.2 bfd fast-detect
minimum-interval 250
RP/0/RP0/CPU0:router(config-static-afi)#10.2.2.0/24 172.16.2.2 bfd fast-detect
minimum-interval 250
RP/0/RP0/CPU0:router(config-static-afi)#10.2.3.0/24 172.16.3.2 bfd fast-detect
minimum-interval 250
RP/0/RP0/CPU0:router(config-static-afi)#exit
RP/0/RP0/CPU0:router(config-static)#exit
!

```

BFD Over Bridge Group Virtual Interface: Example

The following examples show the configurations of the peer and uut nodes. You can see the BVI interface is under a VRF instead of default table:

```

interface BVI100
vrf cctv1 <<<<<<<<<

```

Below is the peer nodes example:

```

l2vpn
bridge group bg
bridge-domain bd
interface Bundle-Ether1.100
!
routed interface BVI100
!
!
!
router vrrp
interface BVI100
bfd minimum-interval 15
address-family ipv4
vrrp 100
address 192.168.1.254
bfd fast-detect peer ipv4 192.168.1.2
!
!
!
router ospf 100
vrf cctv1
router-id 192.168.1.1
area 0
interface BVI100
!
!

```



```

!
!
interface BVI100
 vrf cctv1
  ipv4 address 192.168.1.1 255.255.255.0
!
interface GigE0/1/0/10
 bundle id 1 mode active
 no shut
!
interface Bundle-Ether1
 no shut
!
interface Bundle-Ether1.100 l2transport
 encapsulation dot1q 100
 rewrite ingress tag pop 1 symmetric

!
bfd multipath include loc 0/1/cpu0

interface MgmtEth0/RSP1/CPU0/0
 ipv4 address 7.37.19.20 255.255.0.0
 no shutdown
!
router static
 address-family ipv4 unicast
  0.0.0.0/0 7.37.0.1

```

Below is the uut node example:

```

l2vpn
 bridge group bg
  bridge-domain bd
  interface Bundle-Ether1.100
  !
  routed interface BVI100
  !
!
!
router vrrp
 interface BVI100
  bfd minimum-interval 15
  address-family ipv4
  vrrp 100
  address 192.168.1.254
  bfd fast-detect peer ipv4 192.168.1.1
  !
!
!
router ospf 100
 vrf cctv1
  router-id 192.168.1.2
  area 0
  interface BVI100
  !
!
!
interface BVI100
 vrf cctv1
 ipv4 address 192.168.1.2 255.255.255.0

```

```

!

interface GigE0/1/0/0
  bundle id 1 mode active
  no shut
!
interface Bundle-Ether1
  no shut
!
interface Bundle-Ether1.100 l2transport
  encapsulation dot1q 100
  rewrite ingress tag pop 1 symmetric

!
bfd multipath include location 0/1/CPU0

```

BFD on Bundle Member Links: Examples

The following example shows how to configure BFD on member links of a POS bundle interface:

```

interface Bundle-POS 1
  bfd address-family ipv4 timers start 60
  bfd address-family ipv4 timers nbr-unconfig 60
  bfd address-family ipv4 multiplier 4
  bfd address-family ipv4 destination 192.168.77.2
  bfd address-family ipv4 fast-detect
  bfd address-family ipv4 minimum-interval 120
  ipv4 address 192.168.77.1 255.255.255.252
  ipv6 address babe:cace:cafe:77::1/126
  bundle minimum-active links 2
  bundle minimum-active bandwidth 150000
!
interface Loopback1
  ipv4 address 10.1.1.2 255.255.255.255
!
!
interface Pos0/2/0/0
  bundle id 1 mode active
!
interface Pos0/1/0/0
  bundle id 1 mode active
!
interface Pos0/1/0/1
  bundle id 1 mode active

interface Pos0/1/0/2
  bundle id 1 mode active

interface Pos0/1/0/3
  bundle id 1 mode active
router static
  address-family ipv4 unicast
    ! IPv4 Bundle-Pos1 session, shares ownership with bundle manager
    192.168.177.1/32 192.168.77.2 bfd fast-detect

  address-family ipv6 unicast
    ! Global IPv6 Bundle-Pos1 - Inheriting BFD IPv4 session state
    babe:cace:cafe:177::1/128 babe:cace:cafe:77::2 bfd fast-detect

router ospf foo

```

```

bfd fast-detect
redistribute connected
area 0
  interface Bundle-Pos1
    ! IPv4 Bundle-Pos1 session, shares ownership with bundle manager
    !
router ospfv3 bar
router-id 10.1.1.2
bfd fast-detect
redistribute connected
area 0
  interface Bundle-Pos1
! Linklocal IPv6 Bundle-Pos1 - BFD IPv4 session state

```

The following example shows how to configure BFD on member links of Ethernet bundle interfaces:

```

bfd
interface Bundle-Ether4
  echo disable
  !
interface GigabitEthernet0/0/0/2.3
  echo disable
  !
!
interface GigabitEthernet0/0/0/3 bundle id 1 mode active
interface GigabitEthernet0/0/0/4 bundle id 2 mode active
interface GigabitEthernet0/1/0/2 bundle id 3 mode active
interface GigabitEthernet0/1/0/3 bundle id 4 mode active
interface Bundle-Ether1
  ipv4 address 192.168.1.1/30
  bundle minimum-active links 1
  !
interface Bundle-Ether1.1
  ipv4 address 192.168.100.1/30
  encapsulation dot1q 1001
  !
interface Bundle-Ether2
  bfd address-family ipv4 destination 192.168.2.2
  bfd address-family ipv4 fast-detect
  bfd address-family ipv4 min 83
  bfd address-family ipv4 mul 3
  ipv4 address 192.168.2.1/30
  bundle minimum-active links 1
  !
interface Bundle-Ether3
  bfd address-family ipv4 destination 192.168.3.2
  bfd address-family ipv4 fast-detect
  bfd address-family ipv4 min 83
  bfd address-family ipv4 mul 3
  ipv4 address 192.168.3.1/30
  bundle minimum-active links 1
  !
interface Bundle-Ether4
  bfd address-family ipv4 destination 192.168.4.2
  bfd address-family ipv4 fast-detect
  bfd address-family ipv4 min 83
  bfd address-family ipv4 mul 3
  ipv4 address 192.168.4.1/30
  bundle minimum-active links 1
  !
interface GigabitEthernet 0/0/0/2
  ipv4 address 192.168.10.1/30
  !

```

```

interface GigabitEthernet 0/0/0/2.1
  ipv4 address 192.168.11.1/30
  ipv6 address beef:cafe::1/64
  encapsulation dot1q 2001
!
interface GigabitEthernet 0/0/0/2.2
  ipv4 address 192.168.12.1/30
  encapsulation dot1q 2002
!
interface GigabitEthernet 0/0/0/2.3
  ipv4 address 192.168.13.1/30
  encapsulation dot1q 2003
!
router static
  address-family ipv4 unicast
    10.10.11.2/32 192.168.11.2 bfd fast-detect minimum-interval 250 multiplier 3
    10.10.12.2/32 192.168.12.2 bfd fast-detect minimum-interval 250 multiplier 3
    10.10.13.2/32 192.168.13.2 bfd fast-detect minimum-interval 250 multiplier 3
    10.10.100.2/32 192.168.100.2 bfd fast-detect minimum-interval 250 multiplier 3
!
  address-family ipv6 unicast
    babe:cace::2/128 beef:cafe::2 bfd fast-detect minimum-interval 250 multiplier 3
!

```

Echo Packet Source Address: Examples

The following example shows how to specify the IP address 10.10.10.1 as the source address for BFD echo packets for all BFD sessions on the router:

```

RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router (config)#bfd
RP/0/RP0/CPU0:router (config-bfd)#echo ipv4 source 10.10.10.1

```

The following example shows how to specify the IP address 10.10.10.1 as the source address for BFD echo packets on an individual Gigabit Ethernet interface:

```

RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router (config)#bfd
RP/0/RP0/CPU0:router (config-bfd)#interface gigabitethernet 0/1/0/0
RP/0/RP0/CPU0:router (config-bfd-if)#echo ipv4 source 10.10.10.1

```

The following example shows how to specify the IP address 10.10.10.1 as the source address for BFD echo packets on an individual Packet-over-SONET (POS) interface:

```

RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router (config)#bfd
RP/0/RP0/CPU0:router (config-bfd)#interface pos 0/1/0/0
RP/0/RP0/CPU0:router (config-bfd-if)#echo ipv4 source 10.10.10.1

```

Echo Latency Detection: Examples

In the following examples, consider that the BFD minimum interval is 50 ms, and the multiplier is 3 for the BFD session.

The following example shows how to enable echo latency detection using the default values of 100% of the echo failure period (I x M) for a packet count of 1. In this example, when one echo packet is detected with a roundtrip delay greater than 150 ms, the session is taken down:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#bfd
RP/0/RP0/CPU0:router(config-bfd)#echo latency detect
```

The following example shows how to enable echo latency detection based on 200% (two times) of the echo failure period for a packet count of 1. In this example, when one packet is detected with a roundtrip delay greater than 300 ms, the session is taken down:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#bfd
RP/0/RP0/CPU0:router(config-bfd)#echo latency detect percentage 200
```

The following example shows how to enable echo latency detection based on 100% of the echo failure period for a packet count of 3. In this example, when three consecutive echo packets are detected with a roundtrip delay greater than 150 ms, the session is taken down:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#bfd
RP/0/RP0/CPU0:router(config-bfd)#echo latency detect percentage 100 count 3
```

Echo Startup Validation: Examples

The following example shows how to enable echo startup validation for BFD sessions on non-bundle interfaces if the last received control packet contains a non-zero “Required Min Echo RX Interval” value:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#bfd
RP/0/RP0/CPU0:router(config-bfd)#echo startup validate
```

The following example shows how to enable echo startup validation for BFD sessions on non-bundle interfaces regardless of the “Required Min Echo RX Interval” value in the last control packet:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#bfd
RP/0/RP0/CPU0:router(config-bfd)#echo startup validate force
```

BFD Echo Mode Disable: Examples

The following example shows how to disable echo mode on a router:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#bfd
RP/0/RP0/CPU0:router(config-bfd)#echo disable
```

The following example shows how to disable echo mode on an interface:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router (config)#bfd
RP/0/RP0/CPU0:router (config-bfd)#interface gigabitethernet 0/1/0/0
RP/0/RP0/CPU0:router (config-bfd-if)#echo disable
```

BFD Dampening: Examples

The following example shows how to configure an initial and maximum delay for BFD session startup on BFD bundle members:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router (config)#bfd
RP/0/RP0/CPU0:router (config-bfd)#dampening bundle-member initial-wait 8000
RP/0/RP0/CPU0:router (config-bfd)#dampening bundle-member maximum-wait 15000
```

The following example shows how to change the default initial-wait for BFD on a non-bundle interface:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router (config)#bfd
RP/0/RP0/CPU0:router (config-bfd)#dampening initial-wait 30000
RP/0/RP0/CPU0:router (config-bfd)#dampening maximum-wait 35000
```

BFD IPv6 Checksum: Examples

The following example shows how to disable IPv6 checksum calculations for UDP packets for all BFD sessions on the router:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router (config)#bfd
RP/0/RP0/CPU0:router (config-bfd)#ipv6 checksum disable
```

The following example shows how to reenable IPv6 checksum calculations for UDP packets for all BFD sessions on the router:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router (config)#bfd
RP/0/RP0/CPU0:router (config-bfd)#no ipv6 checksum disable
```

The following example shows how to enable echo mode for BFD sessions on an individual interface:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router (config)#bfd
RP/0/RP0/CPU0:router (config-bfd)#interface gigabitethernet 0/1/0/0
RP/0/RP0/CPU0:router (config-bfd-if)#ipv6 checksum
```

The following example shows how to disable echo mode for BFD sessions on an individual interface:

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router (config)#bfd
RP/0/RP0/CPU0:router (config-bfd)#interface gigabitethernet 0/1/0/0
RP/0/RP0/CPU0:router (config-bfd-if)#ipv6 checksum disable
```

BFD Peers on Routers Running Cisco IOS and Cisco IOS XR Software: Example

The following example shows how to configure BFD on a router interface on Router 1 that is running Cisco IOS software, and use the **bfd neighbor** command to designate the IP address 192.0.2.1 of an interface as its BFD peer on Router 2. Router 2 is running Cisco IOS XR software and uses the **router static** command and **address-family ipv4 unicast** command to designate the path back to Router 1's interface with IP address 192.0.2.2.

Router 1 (Cisco IOS software)

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#interface GigabitEthernet8/1/0
RP/0/RP0/CPU0:router(config-if)#description to-TestBed1 G0/0/0/0
RP/0/RP0/CPU0:router(config-if)#ip address 192.0.2.2 255.255.255.0
RP/0/RP0/CPU0:router(config-if)#bfd interval 100 min_rx 100 multiplier 3
RP/0/RP0/CPU0:router(config-if)#bfd neighbor 192.0.2.1
```

Router 2 (Cisco IOS XR Software)

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#router static
RP/0/RP0/CPU0:router(config-static)#address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-static-afi)#10.10.10.10/32 192.0.2.2 bfd fast-detect
RP/0/RP0/CPU0:router(config-static-afi)#exit
RP/0/RP0/CPU0:router(config-static)#exit
RP/0/RP0/CPU0:router(config)#interface GigabitEthernet0/0/0/0
RP/0/RP0/CPU0:router(config-if)#ipv4 address 192.0.2.1 255.255.255.0
```

BFD over MPLS TE LSPs: Examples

These examples explain how to configure BFD over MPLS TE LSPs.

BFD over MPLS TE Tunnel Head-end Configuration: Example

This example shows how to configure BFD over MPLS TE Tunnel at head-end.

```
bfd multipath include loc 0/1/CPU0
mpls oam
interface tunnel-te 1 bfd fast-detect
interface tunnel-te 1
  bfd minimum-interval
  bfd multiplier
  bfd bringup-timeout
  bfd lsp-ping interval 60
  bfd lsp-ping disable
  bfd dampening initial-wait      (default 16000 ms)
  bfd dampening maximum-wait     (default 600000 ms)
  bfd dampening secondary-wait   (default 20000 ms)
logging events bfd-status
```

BFD over MPLS TE Tunnel Tail-end Configuration: Example

This example shows how to configure BFD over MPLS TE Tunnels at tail-end.

```
bfd multipath include loc 0/1/CPU0
mpls oam
mpls traffic-eng bfd lsp tail multiplier 3
mpls traffic-eng bfd lsp tail minimum-interval 100
```

Where to Go Next

BFD is supported over multiple platforms. For more detailed information about these commands, see the related chapters in the corresponding *Cisco IOS XR Routing Command Reference* and *Cisco IOS XR MPLS Command Reference* for your platform at:

http://www.cisco.com/en/US/products/ps5845/prod_command_reference_list.html

- *BGP Commands on Cisco IOS XR Software*
- *IS-IS Commands on Cisco IOS XR Software*
- *OSPF Commands on Cisco IOS XR Software*
- *Static Routing Commands on Cisco IOS XR Software*
- *MPLS Traffic Engineering Commands on Cisco IOS XR Software*

Additional References

The following sections provide references related to implementing BFD for Cisco IOS XR software.

Related Documents

Related Topic	Document Title
BFD commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Routing Command Reference for Cisco CRS Routers</i>
Configuring QoS packet classification	<i>Modular QoS Configuration Guide for Cisco CRS Routers</i>

Standards

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

RFCs

RFCs	Title
rfc5880_bfd_base	<i>Bidirectional Forwarding Detection</i> , June 2010
rfc5881_bfd_ipv4_ipv6	<i>BFD for IPv4 and IPv6 (Single Hop)</i> , June 2010
rfc5883_bfd_multihop	<i>BFD for Multihop Paths</i> , June 2010

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport



CHAPTER 5

Implementing EIGRP

The Enhanced Interior Gateway Routing Protocol (EIGRP) is an enhanced version of IGRP developed by Cisco. This module describes the concepts and tasks you need to implement basic EIGRP configuration using Cisco IOS XR software. EIGRP uses distance vector routing technology, which specifies that a router need not know all the router and link relationships for the entire network. Each router advertises destinations with a corresponding distance and upon receiving routes, adjusts the distance and propagates the information to neighboring routes.

For EIGRP configuration information related to the following features, see the [Related Documents, on page 345](#) section of this module.

- Multiprotocol Label Switching (MPLS) Layer 3 Virtual Private Network (VPN)
- Site of Origin (SoO) Support



Note For more information about EIGRP on the Cisco IOS XR software and complete descriptions of the EIGRP commands listed in this module, see the *EIGRP Commands* chapter in the *Routing Command Reference for Cisco CRS Routers*. To locate documentation for other commands that might appear while executing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing EIGRP

Release	Modification
Release 3.3.0	This feature was introduced.
Release 3.4.0	Four-byte autonomous system (AS) number support was added.
Release 3.5.0	IPv6 address family and IPv6 VPN routing and forwarding (VRF) address family support was added. IPv6 Provider Edge and IPv6 VPN Provider Edge Transport over Multiprotocol Label Switching Infrastructure support was added.
Release 3.8.0	EIGRP Authentication Keychain feature was added.
Release 4.3.0	Wide Metric Support feature was added

- [Prerequisites for Implementing EIGRP, on page 310](#)
- [Restrictions for Implementing EIGRP , on page 310](#)
- [Information About Implementing EIGRP, on page 310](#)
- [How to Implement EIGRP , on page 323](#)
- [Configuration Examples for Implementing EIGRP , on page 343](#)
- [Additional References, on page 345](#)

Prerequisites for Implementing EIGRP

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.

Restrictions for Implementing EIGRP

The following restrictions are employed when running EIGRP on this version of Cisco IOS XR software:

-
- The characters allowed for EIGRP process name are @ . # : - _ only.
- Simple Network Management Protocol (SNMP) MIB is not supported.
- Interface static routes are not automatically redistributed into EIGRP, because there are no network commands.
- Metric configuration (either through the **default-metric** command or a route policy) is required for redistribution of connected and static routes.
- Auto summary is disabled by default.
- Stub leak maps are not supported.

Information About Implementing EIGRP

To implement EIGRP, you need to understand the following concepts:

EIGRP Functional Overview

Enhanced Interior Gateway Routing Protocol (EIGRP) is an interior gateway protocol suited for many different topologies and media. EIGRP scales well and provides extremely quick convergence times with minimal network traffic.

EIGRP has very low usage of network resources during normal operation. Only hello packets are transmitted on a stable network. When a change in topology occurs, only the routing table changes are propagated and not the entire routing table. Propagation reduces the amount of load the routing protocol itself places on the network. EIGRP also provides rapid convergence times for changes in the network topology.

The distance information in EIGRP is represented as a composite of available bandwidth, delay, load utilization, and link reliability with improved convergence properties and operating efficiency. The fine-tuning of link characteristics achieves optimal paths.

The convergence technology that EIGRP uses is based on research conducted at SRI International and employs an algorithm referred to as the Diffusing Update Algorithm (DUAL). This algorithm guarantees loop-free operation at every instant throughout a route computation and allows all devices involved in a topology change to synchronize at the same time. Routers that are not affected by topology changes are not involved in recomputations. The convergence time with DUAL rivals that of any other existing routing protocol.

EIGRP Features

EIGRP offers the following features:

- Fast convergence—The DUAL algorithm allows routing information to converge as quickly as any currently available routing protocol.
- Partial updates—EIGRP sends incremental updates when the state of a destination changes, instead of sending the entire contents of the routing table. This feature minimizes the bandwidth required for EIGRP packets.
- Neighbor discovery mechanism—This is a simple hello mechanism used to learn about neighboring routers. It is protocol independent.
- Variable-length subnet masks (VLSMs).
- Arbitrary route summarization.
- Scaling—EIGRP scales to large networks.

The following key features are supported in the Cisco IOS XR implementation:

- Support for IPv4 and IPv6 address families.
- Provider Edge (PE)-Customer Edge (CE) protocol support with Site of Origin (SoO) and Border Gateway Protocol (BGP) cost community support.
- PECE protocol support for MPLS and L2TPv3-based-IP L3VPNs.

EIGRP Components

EIGRP has the following four basic components:

- Neighbor discovery or neighbor recovery
- Reliable transport protocol
- DUAL finite state machine
- Protocol-dependent modules

Neighbor discovery or neighbor recovery is the process that routers use to dynamically learn of other routers on their directly attached networks. Routers must also discover when their neighbors become unreachable or inoperative. Neighbor discovery or neighbor recovery is achieved with low overhead by periodically sending small hello packets. As long as hello packets are received, the Cisco IOS XR software can determine that a

neighbor is alive and functioning. After this status is determined, the neighboring routers can exchange routing information.

The reliable transport protocol is responsible for guaranteed, ordered delivery of EIGRP packets to all neighbors. It supports intermixed transmission of multicast and unicast packets. Some EIGRP packets must be sent reliably and others need not be. For efficiency, reliability is provided only when necessary. For example, on a multiaccess network that has multicast capabilities (such as Ethernet) it is not necessary to send hello packets reliably to all neighbors individually. Therefore, EIGRP sends a single multicast hello with an indication in the packet informing the receivers that the packet need not be acknowledged. Other types of packets (such as updates) require acknowledgment, which is indicated in the packet. The reliable transport has a provision to send multicast packets quickly when unacknowledged packets are pending. This provision helps to ensure that convergence time remains low in the presence of various speed links.

The DUAL finite state machine embodies the decision process for all route computations. It tracks all routes advertised by all neighbors. DUAL uses the distance information (known as a metric) to select efficient, loop-free paths. DUAL selects routes to be inserted into a routing table based on a calculation of the feasibility condition. A successor is a neighboring router used for packet forwarding that has a least-cost path to a destination that is guaranteed not to be part of a routing loop. When there are no feasible successors but there are neighbors advertising the destination, a recomputation must occur. This is the process whereby a new successor is determined. The amount of time required to recompute the route affects the convergence time. Recomputation is processor intensive; it is advantageous to avoid unneeded recomputation. When a topology change occurs, DUAL tests for feasible successors. If there are feasible successors, it uses any it finds to avoid unnecessary recomputation.

The protocol-dependent modules are responsible for network layer protocol-specific tasks. An example is the EIGRP module, which is responsible for sending and receiving EIGRP packets that are encapsulated in IP. It is also responsible for parsing EIGRP packets and informing DUAL of the new information received. EIGRP asks DUAL to make routing decisions, but the results are stored in the IP routing table. EIGRP is also responsible for redistributing routes learned by other IP routing protocols.

EIGRP Configuration Grouping

Cisco IOS XR software groups all EIGRP configuration under router EIGRP configuration mode, including interface configuration portions associated with EIGRP. To display EIGRP configuration in its entirety, use the **show running-config router eigrp** command. The command output displays the running configuration for the configured EIGRP instance, including the interface assignments and interface attributes.

EIGRP Configuration Modes

The following examples show how to enter each of the 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:

VRF Configuration Mode

The following example shows how to enter VRF configuration mode:

```
RP/0/RP0/CPU0:router# configuration
RP/0/RP0/CPU0:router(config)# router eigrp 100
```

```
RP/0/RP0/CPU0:router(config-eigrp)# vrf customer1
RP/0/RP0/CPU0:router(config-eigrp-vrf)#
```

IPv4 Address Family Configuration Mode

The following example shows how to enter IPv4 address family configuration mode:

```
RP/0/RP0/CPU0:router# configuration
RP/0/RP0/CPU0:router(config)# router eigrp 100
RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4
RP/0/RP0/CPU0:router(config-eigrp-af)#
```

IPv6 Address Family Configuration Mode

The following example shows how to enter IPv6 address family configuration mode:

```
RP/0/RP0/CPU0:router# configuration
RP/0/RP0/CPU0:router(config)# router eigrp 100
RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv6
RP/0/RP0/CPU0:router(config-eigrp-af)#
```

IPv4 VRF Address Family Configuration Mode

The following example shows how to enter IPv4 VRF address family configuration mode:

```
RP/0/RP0/CPU0:router# configuration
RP/0/RP0/CPU0:router(config)# router eigrp 100
RP/0/RP0/CPU0:router(config-eigrp)# vrf customer1
RP/0/RP0/CPU0:router(config-eigrp-vrf)# address-family ipv4
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)#
```

IPv6 VRF Address Family Configuration Mode

The following example shows how to enter IPv6 VRF address family configuration mode:

```
RP/0/RP0/CPU0:router# configuration
RP/0/RP0/CPU0:router(config)# router eigrp 100
RP/0/RP0/CPU0:router(config-eigrp)# vrf customer1
RP/0/RP0/CPU0:router(config-eigrp-vrf)# address-family ipv6
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)#
```

Interface Configuration Mode

The following example shows how to enter interface configuration mode in IPv4 address family configuration mode:

```
RP/0/RP0/CPU0:router# configuration
RP/0/RP0/CPU0:router(config)# router eigrp 100
RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4
RP/0/RP0/CPU0:router(config-eigrp-af)# interface GigabitEthernet 0/3/0/0
RP/0/RP0/CPU0:router(config-eigrp-af-if)#
```

The following example shows how to enter interface configuration mode in IPv6 VRF configuration mode:

```
RP/0/RP0/CPU0:router# configuration
RP/0/RP0/CPU0:router(config)# router eigrp 100
RP/0/RP0/CPU0:router(config-eigrp)# vrf customer1
RP/0/RP0/CPU0:router(config-eigrp-vrf)# address-family ipv6
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# interface POS0/5/0/0
RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)#
```

EIGRP Interfaces

EIGRP interfaces can be configured as either of the following types:

- **Active**—Advertises connected prefixes and forms adjacencies. This is the default type for interfaces.
- **Passive**—Advertises connected prefixes but does not form adjacencies. The **passive** command is used to configure interfaces as passive. Passive interfaces should be used sparingly for important prefixes, such as loopback addresses, that need to be injected into the EIGRP domain. If many connected prefixes need to be advertised, then the redistribution of connected routes with the appropriate policy should be used instead.

Redistribution for an EIGRP Process

Routes from other protocols can be redistributed into EIGRP. A route policy can be configured along with the **redistribute** command. A metric is required, configured either through the **default-metric** command or under the route policy configured with the **redistribute** command to import routes into EIGRP.

A route policy allows the filtering of routes based on attributes such as the destination, origination protocol, route type, route tag, and so on. When redistribution is configured under a VRF, EIGRP retrieves extended communities attached to the route in the routing information base (RIB). The SoO is used to filter out routing loops in the presence of MPLS VPN backdoor links.

Metric Weights for EIGRP Routing

EIGRP uses the minimum bandwidth on the path to a destination network and the total delay to compute routing metrics. You can use the **metric weights** command to adjust the default behavior of EIGRP routing and metric computations. For example, this adjustment allows you to tune system behavior to allow for satellite transmission. EIGRP metric defaults have been carefully selected to provide optimal performance in most networks.

By default, the EIGRP composite metric is a 32-bit quantity that is a sum of the segment delays and lowest segment bandwidth (scaled and inverted) for a given route. For a network of homogeneous media, this metric reduces to a hop count. For a network of mixed media (FDDI, Ethernet, and serial lines running from 9600 bits per second to T1 rates), the route with the lowest metric reflects the most desirable path to a destination.

Mismatched K Values

Mismatched K values (EIGRP metrics) can prevent neighbor relationships from being established and can negatively impact network convergence. The following example explains this behavior between two EIGRP peers (ROUTER-A and ROUTER-B).

The following error message is displayed in the console of ROUTER-B because the K values are mismatched:


```
RP/0/RP0/CPU0:Mar 13 08:19:55:eigrp[163]:%ROUTING-EIGRP-5-NBRCHANGE:IP-EIGRP(0) 1:Neighbor
11.0.0.20 (GigabitEthernet0/6/0/0) is down: K-value mismatch
```

Two scenarios occur in which this error message can be displayed:

- The two routers are connected on the same link and configured to establish a neighbor relationship. However, each router is configured with different K values.

The following configuration is applied to ROUTER-A. The K values are changed with the **metric weights** command. A value of 2 is entered for the *k1* argument to adjust the bandwidth calculation. The value of 1 is entered for the *k3* argument to adjust the delay calculation.

```
hostname ROUTER-A!
interface GigabitEthernet0/6/0/0
  ipv4 address 10.1.1.1 255.255.255.0

router eigrp 100
  metric weights 0 2 0 1 0 0
  interface GigabitEthernet0/6/0/0
```

The following configuration is applied to ROUTER-B. However, the **metric weights** command is not applied and the default K values are used. The default K values are 1, 0, 1, 0, and 0.

```
hostname ROUTER-B!
interface GigabitEthernet0/6/0/1
  ipv4 address 10.1.1.2 255.255.255.0

router eigrp 100
  interface GigabitEthernet0/6/0/1
```

The bandwidth calculation is set to 2 on ROUTER-A and set to 1 (by default) on ROUTER-B. This configuration prevents these peers from forming a neighbor relationship.

- The K-value mismatch error message can also be displayed if one of the two peers has transmitted a “goodbye” message and the receiving router does not support this message. In this case, the receiving router interprets this message as a K-value mismatch.

Goodbye Message

The goodbye message is a feature designed to improve EIGRP network convergence. The goodbye message is broadcast when an EIGRP routing process is shut down to inform adjacent peers about the impending topology change. This feature allows supporting EIGRP peers to synchronize and recalculate neighbor relationships more efficiently than would occur if the peers discovered the topology change after the hold timer expired.

The following message is displayed by routers that run a supported release when a goodbye message is received:

```
RP/0/RP0/CPU0:Mar 13 09:13:17:eigrp[163]:%ROUTING-EIGRP-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor
10.0.0.20 (GigabitEthernet0/6/0/0) is down: Interface Goodbye received
```

A Cisco router that runs a software release that does not support the goodbye message can misinterpret the message as a K-value mismatch and display the following message:

```
RP/0/RP0/CPU0:Mar 13 09:13:17:eigrp[163]:%ROUTING-EIGRP-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor
10.0.0.20 (GigabitEthernet0/6/0/0) is down: K-value mismatch
```



Note The receipt of a goodbye message by a nonsupporting peer does not disrupt normal network operation. The nonsupporting peer terminates the session when the hold timer expires. The sending and receiving routers reconverge normally after the sender reloads.

Percentage of Link Bandwidth Used for EIGRP Packets

By default, EIGRP packets consume a maximum of 50 percent of the link bandwidth, as configured with the **bandwidth** interface configuration command. You might want to change that value if a different level of link utilization is required or if the configured bandwidth does not match the actual link bandwidth (it may have been configured to influence route metric calculations).

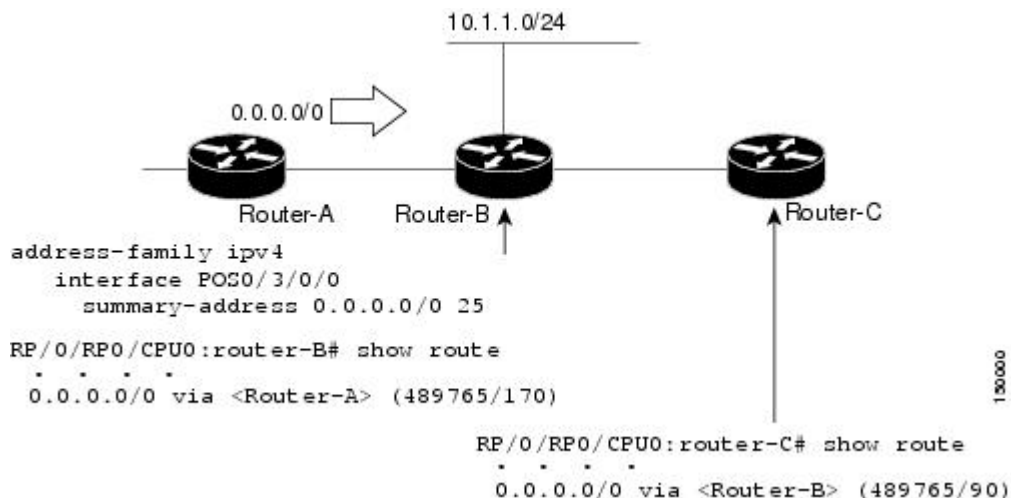
Floating Summary Routes for an EIGRP Process

You can also use a floating summary route when configuring the **summary-address** command. The floating summary route is created by applying a default route and administrative distance at the interface level. The following scenario illustrates the behavior of this enhancement.

[Figure 17: Floating Summary Route Is Applied to Router-B, on page 316](#) shows a network with three routers, Router-A, Router-B, and Router-C. Router-A learns a default route from elsewhere in the network and then advertises this route to Router-B. Router-B is configured so that only a default summary route is advertised to Router-C. The default summary route is applied to interface 0/1 on Router-B with the following configuration:

```
RP/0/RP0/CPU0:router(config)# router eigrp 100
RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4
RP/0/RP0/CPU0:router(config-eigrp-af)# interface GigabitEthernet 0/3/0/0
RP/0/RP0/CPU0:router(config-eigrp-af-if)# summary-address 100.0.0.0 0.0.0.0
```

Figure 17: Floating Summary Route Is Applied to Router-B



The configuration of the default summary route on Router-B sends a 0.0.0.0/0 summary route to Router-C and blocks all other routes, including the 10.1.1.0/24 route, from being advertised to Router-C. However, this configuration also generates a local discard route on Router-B, a route for 0.0.0.0/0 to the null 0 interface with

an administrative distance of 5. When this route is created, it overrides the EIGRP learned default route. Router-B is no longer able to reach destinations that it would normally reach through the 0.0.0.0/0 route.

This problem is resolved by applying a floating summary route to the interface on Router-B that connects to Router-C. The floating summary route is applied by relating an administrative distance to the default summary route on the interface of Router-B with the following statement:

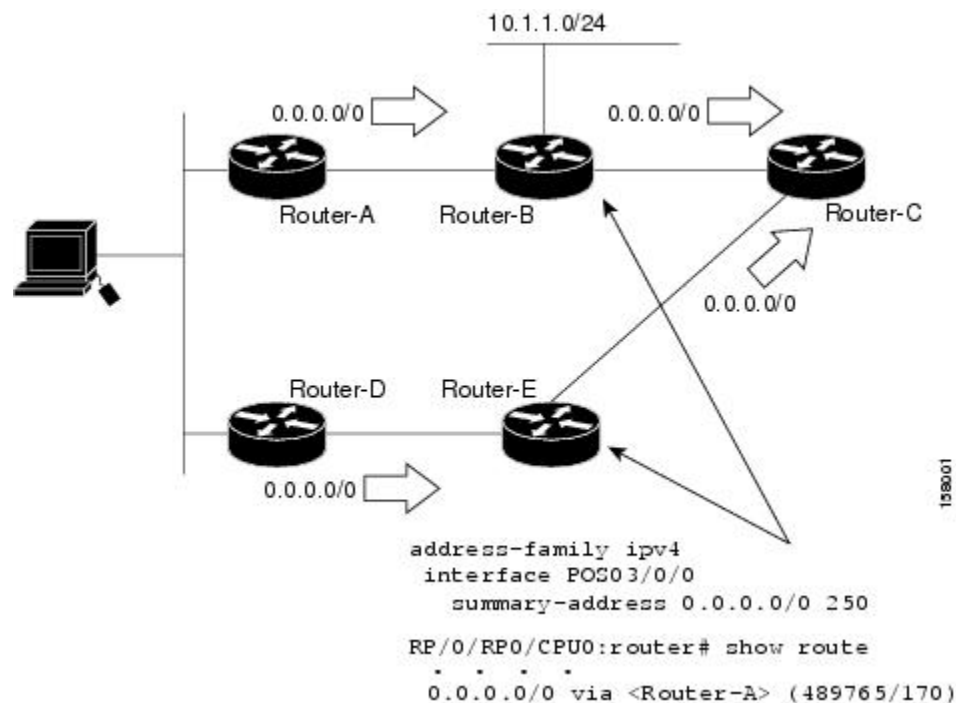
```
RP/0/RP0/CPU0:router(config-if)# summary-address 100 0.0.0.0 0.0.0.0 250
```

The administrative distance of 250, applied in the above statement, is now assigned to the discard route generated on Router-B. The 0.0.0.0/0, from Router-A, is learned through EIGRP and installed in the local routing table. Routing to Router-C is restored.

If Router-A loses the connection to Router-B, Router-B continues to advertise a default route to Router-C, which allows traffic to continue to reach destinations attached to Router-B. However, traffic destined for networks to Router-A or behind Router-A is dropped when the traffic reaches Router-B.

[Figure 18: Floating Summary Route Applied for Dual-Homed Remotes, on page 317](#) shows a network with two connections from the core: Router-A and Router-D. Both routers have floating summary routes configured on the interfaces connected to Router-C. If the connection between Router-E and Router-C fails, the network continues to operate normally. All traffic flows from Router-C through Router-B to the hosts attached to Router-A and Router-D.

Figure 18: Floating Summary Route Applied for Dual-Homed Remotes



However, if the link between Router-D and Router-E fails, the network may dump traffic into a black hole because Router-E continues to advertise the default route (0.0.0.0/0) to Router-C, as long as at least one link (other than the link to Router-C) to Router-E is still active. In this scenario, Router-C still forwards traffic to Router-E, but Router-E drops the traffic creating the black hole. To avoid this problem, you should configure

the summary address with an administrative distance on only single-homed remote routers or areas in which only one exit point exists between the segments of the network. If two or more exit points exist (from one segment of the network to another), configuring the floating default route can cause a black hole to form.

Split Horizon for an EIGRP Process

Split horizon controls the sending of EIGRP update and query packets. When split horizon is enabled on an interface, update and query packets are not sent for destinations for which this interface is the next hop. Controlling update and query packets in this manner reduces the possibility of routing loops.

By default, split horizon is enabled on all interfaces.

Split horizon blocks route information from being advertised by a router on any interface from which that information originated. This behavior usually optimizes communications among multiple routing devices, particularly when links are broken. However, with nonbroadcast networks (such as Frame Relay and SMDS), situations can arise for which this behavior is less than ideal. For these situations, including networks in which you have EIGRP configured, you may want to disable split horizon.

Adjustment of Hello Interval and Hold Time for an EIGRP Process

You can adjust the interval between hello packets and the hold time.

Routing devices periodically send hello packets to each other to dynamically learn of other routers on their directly attached networks. This information is used to discover neighbors and learn when neighbors become unreachable or inoperative. By default, hello packets are sent every 5 seconds.

You can configure the hold time on a specified interface for a particular EIGRP routing process designated by the autonomous system number. The hold time is advertised in hello packets and indicates to neighbors the length of time they should consider the sender valid. The default hold time is three times the hello interval, or 15 seconds.

Stub Routing for an EIGRP Process

The EIGRP Stub Routing feature improves network stability, reduces resource usage, and simplifies stub router configuration.

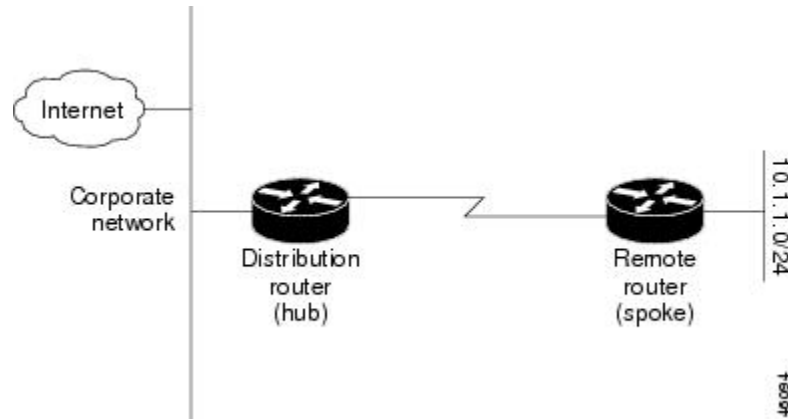
Stub routing is commonly used in a hub-and-spoke network topology. In a hub-and-spoke network, one or more end (stub) networks are connected to a remote router (the spoke) that is connected to one or more distribution routers (the hub). The remote router is adjacent only to one or more distribution routers. The only route for IP traffic to follow into the remote router is through a distribution router. This type of configuration is commonly used in WAN topologies in which the distribution router is directly connected to a WAN. The distribution router can be connected to many more remote routers. Often, the distribution router is connected to 100 or more remote routers. In a hub-and-spoke topology, the remote router must forward all nonlocal traffic to a distribution router, so it becomes unnecessary for the remote router to hold a complete routing table. Generally, the distribution router need not send anything more than a default route to the remote router.

When using the EIGRP Stub Routing feature, you need to configure the distribution and remote routers to use EIGRP and configure only the remote router as a stub. Only specified routes are propagated from the remote (stub) router. The stub router responds to all queries for summaries, connected routes, redistributed static routes, external routes, and internal routes with the message “inaccessible.” A router that is configured as a stub sends a special peer information packet to all neighboring routers to report its status as a stub router.

Any neighbor that receives a packet informing it of the stub status does not query the stub router for any routes, and a router that has a stub peer does not query that peer. The stub router depends on the distribution router to send the proper updates to all peers.

Figure 19: Simple Hub-and-Spoke Network

This figure shows a simple hub-and-spoke configuration.



The stub routing feature by itself does not prevent routes from being advertised to the remote router. In the example in [Figure 19: Simple Hub-and-Spoke Network, on page 319](#), the remote router can access the corporate network and the Internet through the distribution router only. Having a full route table on the remote router, in this example, would serve no functional purpose because the path to the corporate network and the Internet would always be through the distribution router. The larger route table would only reduce the amount of memory required by the remote router. Bandwidth and memory can be conserved by summarizing and filtering routes in the distribution router. The remote router need not receive routes that have been learned from other networks because the remote router must send all nonlocal traffic, regardless of destination, to the distribution router. If a true stub network is desired, the distribution router should be configured to send only a default route to the remote router. The EIGRP Stub Routing feature does not automatically enable summarization on the distribution router. In most cases, the network administrator needs to configure summarization on the distribution routers.

Without the stub feature, even after the routes that are sent from the distribution router to the remote router have been filtered or summarized, a problem might occur. If a route is lost somewhere in the corporate network, EIGRP could send a query to the distribution router, which in turn sends a query to the remote router even if routes are being summarized. If there is a problem communicating over the WAN link between the distribution router and the remote router, an EIGRP stuck in active (SIA) condition could occur and cause instability elsewhere in the network. The EIGRP Stub Routing feature allows a network administrator to prevent queries from being sent to the remote router.

Route Policy Options for an EIGRP Process

Route policies comprise series of statements and expressions that are bracketed with the **route-policy** and **end-policy** keywords. Rather than a collection of individual commands (one for each line), the statements within a route policy have context relative to each other. Thus, instead of each line being an individual command, each policy or set is an independent configuration object that can be used, entered, and manipulated as a unit.

Each line of a policy configuration is a logical subunit. At least one new line must follow the **then**, **else**, and **end-policy** keywords. A new line must also follow the closing parenthesis of a parameter list and the

name string in a reference to an AS path set, community set, extended community set, or prefix set (in the EIGRP context). At least one new line must precede the definition of a route policy or prefix set. A new line must appear at the end of a logical unit of policy expression and may not appear anywhere else.

This is the command to set the EIGRP metric in a route policy:

```
RP/0/RP0/CPU0:router(config-rpl)# set eigrp-metric bandwidth delay reliability loading mtu
```

This is the command to provide EIGRP offset list functionality in a route policy:

```
RP/0/RP0/CPU0:router(config-rpl)# add eigrp-metric bandwidth delay reliability loading mtu
```

A route policy can be used in EIGRP only if all the statements are applicable to the particular EIGRP attach point. The following commands accept a route policy:

- **default-information allowed**—Match statements are allowed for destination. No set statements are allowed.
- **route-policy**—Match statements are allowed for destination, next hop, and tag. Set statements are allowed for eigrp-metric and tag.
- **redistribute**—Match statements are allowed for destination, next hop, source-protocol, tag and route-type. Set statements are allowed for eigrp-metric and tag.

The range for setting a tag is 0 to 255 for internal routes and 0 to 4294967295 for external routes.

EIGRP Layer 3 VPN PE-CE Site-of-Origin

The EIGRP MPLS and IP VPN PE-CE Site-of-Origin (SoO) feature introduces the capability to filter Multiprotocol Label Switching (MPLS) and IP Virtual Private Network (VPN) traffic on a per-site basis for EIGRP networks. SoO filtering is configured at the interface level and is used to manage MPLS and IP VPN traffic and to prevent transient routing loops from occurring in complex and mixed network topologies.

Router Interoperation with the Site-of-Origin Extended Community

The configuration of the SoO extended community allows routers that support this feature to identify the site from which each route originated. When this feature is enabled, the EIGRP routing process on the PE or CE router checks each received route for the SoO extended community and filters based on the following conditions:

- A received route from BGP or a CE router contains a SoO value that matches the SoO value on the receiving interface:
 - If a route is received with an associated SoO value that matches the SoO value that is configured on the receiving interface, the route is filtered out because it was learned from another PE router or from a backdoor link. This behavior is designed to prevent routing loops.
- A received route from a CE router is configured with a SoO value that does not match:
 - If a route is received with an associated SoO value that does not match the SoO value that is configured on the receiving interface, the route is accepted into the EIGRP topology table so that it can be redistributed into BGP.
 - If the route is already installed in the EIGRP topology table but is associated with a different SoO value, the SoO value from the topology table is used when the route is redistributed into BGP.

- A received route from a CE router does not contain a SoO value:
 - If a route is received without a SoO value, the route is accepted into the EIGRP topology table, and the SoO value from the interface that is used to reach the next-hop CE router is appended to the route before it is redistributed into BGP.

When BGP and EIGRP peers that support the SoO extended community receive these routes, they also receive the associated SoO values and pass them to other BGP and EIGRP peers that support the SoO extended community. This filtering is designed to prevent transient routes from being relearned from the originating site, which prevents transient routing loops from occurring.

In conjunction with BGP cost community, EIGRP, BGP, and the RIB ensure that paths over the MPLS VPN core are preferred over backdoor links.

For MPLS and IP VPN and SoO configuration information, see *Implementing MPLS Layer 3 VPNs* in the *Cisco IOS XR Multiprotocol Label Switching Configuration Guide*.

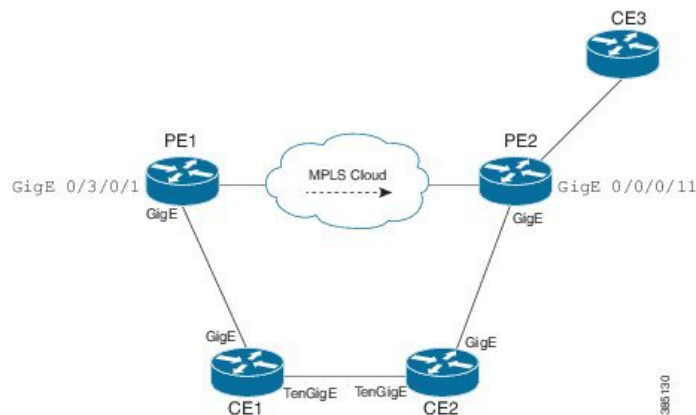
Route Manipulation using SoO match condition

The SoO configuration in EIGRP network can be used to manipulate routes using the SoO match condition in the routing policy. The egress interface of a PE router is used to compare and manipulate routes based on the SoO configuration on the remote PE router.

Topology

In the following topology, CE1, CE2 and CE3 are the customer edge routers. PE1 and PE2 are the provider edge routers. By default, CE1 will use PE1->PE2 to reach CE3. To configure CE1 to use CE2 to reach CE3, the metric advertised by PE1 must be increased.

The routing policy on PE1 manipulates routes received from CE3 via PE2, by using the SoO match condition. With this feature added, PE1 can increase the metric while advertising routes to CE1.



Configuration:

```
/*SoO tag is assigned on PE2 router*/

router(config)#interface GigabitEthernet0/0/0/11
router (config-if)#site-of-origin 33.33.33.33:33

/* A route-policy defined on PE1 */
```

```

router(config)#route-policy test
router(config-rpl)#if extcommunity soo matches-any (33.33.33.33:33) then
router(config-rpl-if)#set eigrp-metric 2121212121 333333333 245 250 1455
router(config-rpl-if)#endif
router(config-rpl)#end-policy
router (config)#commit
router (config)#

router(config)#interface GigabitEthernet0/3/0/1
router (config-if)#route-policy test out

```

Verification:

```

/*A route with poor metric advertised by PE1 is installed into CE1's routing table for SoO
of site C3. */

```

```

router#show eigrp topology 6:6::1/128

```

```

IPv6-EIGRP AS(100): Topology entry for 6:6::1/128
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 15539149614794, RIB is
4294967295 Routing Descriptor Blocks: fe80::226:98ff:fe24:5109 (GigabitEthernet0/0/0/15),
from fe80::226:98ff:fe24:5109, Send flag is 0x0
Composite metric is (15539149614794/15539148304382), Route is Internal Vector metric:
Minimum bandwidth is 1000000 Kbit
Total delay is 237108596182784 picoseconds
Reliability is 245/255
Load is 250/255
Minimum MTU is 1455
Hop count is 2
Originating router is 2.2.2.2
Extended Community:
SoO:33.33.33.33:33

```

Note:

This feature is applicable to both ipv4 as well as ipv6.

All types of SoO(IP-Address, ASN2, ASN4) are supported.

IPv6 and IPv6 VPN Provider Edge Support over MPLS and IP

IPv6 Provider Edge (6PE) and IPv6 VPN Provider Edge (6VPE) uses the existing IP and Multiprotocol Label Switching (MPLS) IPv4 core infrastructure for IPv6 transport. 6PE and 6VPE enable IPv6 sites to communicate with each other over an IP and MPLS IPv4 core network using MPLS label switched paths (LSPs).

EIGRP is an Interior Gateway Protocol (IGP) that supports the 6PE or 6VPE provider edge-to-customer edge protocol by supporting the configuration of IPv6 address families in EIGRP VRF and exchanging IPv6 routing updates in the L3VPN environment

For detailed information on configuring 6PE and 6VPE over MPLS and IP, see *MPLS Configuration Guide for the Cisco CRS Routers*.

EIGRP v4/v6 Authentication Using Keychain

EIGRP authentication using keychain introduces the capability to authenticate EIGRP protocol packets on a per-interface basis. The EIGRP routing authentication provides a mechanism to authenticate all EIGRP protocol traffic on one or more interfaces, based on Message Digest 5 (MD5) authentication.

The EIGRP routing authentication uses the Cisco IOS XR software security keychain infrastructure to store and retrieve secret keys and to authenticate incoming and outgoing traffic on a per-interface basis.

EIGRP Wide Metric Computation

The Cisco IOS XR Enhanced Interior Gateway Routing Protocol (EIGRP) implementation is enhanced to perform wide metric computation. This enhancement is to support high bandwidth interfaces.

A new EIGRP command is added and existing EIGRP commands are enhanced to support wide metric computation feature.

- **metric rib-scale**—This command was introduced.
- **metric**—The **picoseconds** keyword was added.
- **metric weights**—Support was added for the *k6* constant.
- **show eigrp interfaces**—The command output was modified to display relevant wide metric information.
- **show eigrp neighbors**—The command output was modified to display relevant wide metric information.
- **show eigrp topology**—The command output was modified to display relevant wide metric information.
- **show protocols**—The command output was modified to display relevant wide metric information.

**Note**

If there is a combination of IOS and IOS-XR PE devices in the network, then the EIGRP wide metric must be disabled in IOS-XR PE device. This is because the method of calculating metrics in L3VPN design between IOS and IOS-XR.

EIGRP Support for BFD

EIGRP supports Bidirectional forwarding detection (BFD) for link failure detection.

BFD provides low-overhead, short-duration detection of failures in the path between adjacent forwarding engines. BFD allows a single mechanism to be used for failure detection over any media and at any protocol layer, with a wide range of detection times and overhead. The fast detection of failures provides immediate reaction to failure in the event of a failed link or neighbor.

How to Implement EIGRP

This section contains instructions for the following tasks:



Note To save configuration changes, you must commit changes when the system prompts you.

Enabling EIGRP Routing

This task enables EIGRP routing and establishes an EIGRP routing process.

Before you begin

Although you can configure EIGRP before you configure an IP address, no EIGRP routing occurs until at least one IP address is configured.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **address-family** { **ipv4** | **ipv6** }
4. **router-id** *id*
5. **default-metric** *bandwidth delay reliability loading mtu*
6. **distance** *internal-distance external-distance*
7. **interface** *type interface-path-id*
8. **holdtime** *seconds*
9. **bandwidth-percent** *percent*
10. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router eigrp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router eigrp 100	Specifies the autonomous system number of the routing process to configure an EIGRP routing process.
Step 3	address-family { ipv4 ipv6 } Example: RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4	Enters an address family configuration mode.
Step 4	router-id <i>id</i> Example: RP/0/RP0/CPU0:router(config-eigrp)# router-id 172.20.1.1	(Optional) Configures a router-id for an EIGRP process. Note It is good practice to use the router-id command to explicitly specify a unique 32-bit numeric value for the router ID. This action ensures that EIGRP can function regardless of the interface address configuration.

	Command or Action	Purpose
Step 5	<p>default-metric <i>bandwidth delay reliability loading mtu</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-eigrp-af)# default-metric 1000 100 250 100 1500</pre>	(Optional) Sets metrics for an EIGRP process.
Step 6	<p>distance <i>internal-distance external-distance</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-eigrp-af)# distance 80 130</pre>	(Optional) Allows the use of two administrative distances—internal and external—that could be a better route to a node.
Step 7	<p>interface <i>type interface-path-id</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-eigrp-af)# interface GigabitEthernet 0/1/0/0</pre>	Defines the interfaces on which the EIGRP routing protocol runs.
Step 8	<p>holdtime <i>seconds</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-eigrp-af-if)# holdtime 30</pre>	<p>(Optional) Configures the hold time for an interface.</p> <p>Note To ensure nonstop forwarding during RP failovers, as the number of neighbors increase, a higher holdtime than the default value is recommended. With 256 neighbors across all VRFs, we recommend 60 seconds.</p>
Step 9	<p>bandwidth-percent <i>percent</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-eigrp-af-if)# bandwidth-percent 75</pre>	(Optional) Configures the percentage of bandwidth that may be used by EIGRP on an interface.
Step 10	commit	

Configuring Route Summarization for an EIGRP Process

This task configures route summarization for an EIGRP process.

You can configure a summary aggregate address for a specified interface. If any more specific routes are in the routing table, EIGRP advertises the summary address from the interface with a metric equal to the minimum of all more specific routes.

Before you begin

Note You should not use the **summary-address** summarization command to generate the default route (0.0.0.0) from an interface. This command creates an EIGRP summary default route to the null 0 interface with an administrative distance of 5. The low administrative distance of this default route can cause this route to displace default routes learned from other neighbors from the routing table. If the default route learned from the neighbors is displaced by the summary default route or the summary route is the only default route present, all traffic destined for the default route does not leave the router; instead, this traffic is sent to the null 0 interface, where it is dropped.

The recommended way to send only the default route from a given interface is to use a **route-policy** command.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **address-family** { **ipv4** | **ipv6** }
4. **route-policy** *name out*
5. **interface** *type interface-path-id*
6. **summary-address** *ip-address* { */ length* | *mask* } [*admin-distance*]
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router eigrp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router eigrp 100	Specifies the AS number of the routing process to configure an EIGRP routing process.
Step 3	address-family { ipv4 ipv6 } Example: RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4	Enters an address family configuration mode, IPv4 or IPv6.
Step 4	route-policy <i>name out</i> Example: RP/0/RP0/CPU0:router(config-eigrp-af)# route-policy FILTER_DEFAULT out	Applies a routing policy to updates advertised to or received from an EIGRP neighbor.
Step 5	interface <i>type interface-path-id</i> Example:	Defines the interfaces on which the EIGRP routing protocol runs.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-eigrp-af)# interface GigabitEthernet 0/1/0/0	
Step 6	summary-address <i>ip-address</i> { / <i>length</i> <i>mask</i> } [<i>admin-distance</i>] Example: RP/0/RP0/CPU0:router(config-eigrp-af-if)# summary-address 192.168.0.0/16 95	Configures a summary aggregate address for the specified EIGRP interface.
Step 7	commit	

Redistributing Routes for EIGRP

This task explains how to redistribute routes, apply limits on the number of routes, and set timers for nonstop forwarding.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **address-family** { *ipv4* | *ipv6* }
4. **redistribute** {{ *bgp* | *connected* | *isis* | *ospf* | *ospfv3* | *rip* | *static* } [*as-number*]} [*route-policy name*]
5. **redistribute maximum-prefix** *maximum* [*threshold*] [[*dampened*] [*reset-time minutes*] [*restart minutes*] [*restart-count number*] | [*warning-only*]]
6. **timers nsf route-hold** *seconds*
7. **maximum paths** *maximum*
8. **maximum-prefix** *maximum* [*threshold*] [[*dampened*] [*reset-time minutes*] [*restart minutes*] [*restart-count number*] | [*warning-only*]]
9. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router eigrp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router eigrp 100	Specifies the AS number of the routing process to configure an EIGRP routing process.
Step 3	address-family { <i>ipv4</i> <i>ipv6</i> } Example: RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4	Enters an address family configuration mode, IPv4 or IPv6.

	Command or Action	Purpose
Step 4	<p>redistribute { { bgp connected isis ospf ospfv3 rip static } [<i>as-number</i>] } [route-policy <i>name</i>]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-eigrp-af)# redistribute bgp 100</pre>	Redistributes the routes from the specified protocol and AS number to the EIGRP process. Optionally, the redistributed routes can be filtered into the EIGRP process by providing the route policy.
Step 5	<p>redistribute maximum-prefix <i>maximum</i> [<i>threshold</i>] [[dampened] [reset-time <i>minutes</i>] [restart <i>minutes</i>] [restart-count <i>number</i>] [warning-only]]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-eigrp-af)# redistribute maximum-prefix 5000 95 warning-only</pre>	<p>Limits the maximum number of prefixes that are redistributed to the EIGRP process.</p> <p>Caution After the restart count threshold is crossed, you need to use the <code>clear eigrp neighbors</code> command to re-establish normal peering, redistribution, or both.</p>
Step 6	<p>timers nsf route-hold <i>seconds</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-eigrp-af)# timers nsf route-hold 120</pre>	Sets the timer that determines how long an NSF-aware EIGRP router holds routes for an inactive peer.
Step 7	<p>maximum paths <i>maximum</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-eigrp-af)# maximum paths 10</pre>	Controls the maximum number of parallel routes that the EIGRP can support.
Step 8	<p>maximum-prefix <i>maximum</i> [<i>threshold</i>] [[dampened] [reset-time <i>minutes</i>] [restart <i>minutes</i>] [restart-count <i>number</i>] [warning-only]]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-eigrp-af)# maximum-prefix 50000</pre>	Limits the number of prefixes that are accepted under an address family by EIGRP.
Step 9	commit	

Creating a Route Policy and Attaching It to an EIGRP Process

This task defines a route policy and shows how to attach it to an EIGRP process.

A route policy definition consists of the **route-policy** command and *name* argument followed by a sequence of optional policy statements, and then closed with the **end-policy** command.

A route policy is not useful until it is applied to routes of a routing protocol.

SUMMARY STEPS

1. **configure**

2. **route-policy** *name*
3. **set eigrp-metric** *bandwidth delay reliability load mtu*
4. **end-policy**
5. **commit**
6. **configure**
7. **router eigrp** *as-number*
8. **address-family** { **ipv4** | **ipv6** }
9. **route-policy** *route-policy-name* { **in** | **out** }
10. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	route-policy <i>name</i> Example: RP/0/RP0/CPU0:router(config)# route-policy IN-IPv4	Defines a route policy and enters route-policy configuration mode.
Step 3	set eigrp-metric <i>bandwidth delay reliability load mtu</i> Example: RP/0/RP0/CPU0:router(config-rpl)# set eigrp metric 42 100 200 100 1200	(Optional) Sets the EIGRP metric attribute.
Step 4	end-policy Example: RP/0/RP0/CPU0:router(config-rpl)# end-policy	Ends the definition of a route policy and exits route-policy configuration mode.
Step 5	commit	
Step 6	configure	
Step 7	router eigrp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router eigrp 100	Specifies the autonomous system number of the routing process to configure an EIGRP routing process.
Step 8	address-family { ipv4 ipv6 } Example: RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4	Enters an address family configuration mode.
Step 9	route-policy <i>route-policy-name</i> { in out } Example:	Applies a routing policy to updates advertised to or received from an EIGRP neighbor.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-eigrp-af)# route-policy IN-IPv4 in	
Step 10	commit	

Configuring Stub Routing for an EIGRP Process

This task configures the distribution and remote routers to use an EIGRP process for stub routing.

Before you begin



Note EIGRP stub routing should be used only on remote routers. A stub router is defined as a router connected to the network core or distribution layer through which core transit traffic should not flow. A stub router should not have any EIGRP neighbors other than distribution routers. Ignoring this restriction causes undesirable behavior.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **address-family** { **ipv4** | **ipv6** }
4. **stub** [**receive-only** | {[**connected**] [**redistributed**] [**static**] [**summary**]}]
5. **commit**
6. **show eigrp** [**ipv4** | **ipv6**] **neighbors** [*as-number*] [**detail**] [*type interface-path-id* | **static**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router eigrp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router eigrp 100	Specifies the autonomous system number of the routing process to configure an EIGRP routing process.
Step 3	address-family { ipv4 ipv6 } Example: RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4	Enters an address family configuration mode.
Step 4	stub [receive-only {[connected] [redistributed] [static] [summary]}] Example:	Configures a router as a stub for EIGRP.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-eigrp-af)# stub receive-only	
Step 5	commit	
Step 6	show eigrp [ipv4 ipv6] neighbors [as-number] [detail] [type interface-path-id static] Example: RP/0/RP0/CPU0:router# show eigrp neighbors detail	Verifies that a remote router has been configured as a stub router with EIGRP. The last line of the output shows the stub status of the remote or spoke router.

Configuring EIGRP as a PE-CE Protocol

Perform this task to configure EIGRP on the provider edge (PE) and establish provider edge-to-customer edge (PE-CE) communication using EIGRP.

SUMMARY STEPS

1. **configure**
2. **router eigrp as-number**
3. **vrf vrf-name**
4. **address-family { ipv4 | ipv6 }**
5. **router-id router-id**
6. **autonomous-system as-number**
7. **redistribute {{ bgp | connected | isis | ospf | ospfv3 | rip | static } [as-number | instance-name]} [route-policy name]**
8. **interface type interface-path-id**
9. **site-of-origin { as-number:number | ip-address : number }**
10. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router eigrp as-number Example: RP/0/RP0/CPU0:router(config)# router eigrp 100	Specifies the autonomous system number of the routing process to configure an EIGRP routing process.
Step 3	vrf vrf-name Example: RP/0/RP0/CPU0:router(config-eigrp)# vrf vrf_A	Configures a VPN routing and forwarding (VRF) instance.

	Command or Action	Purpose
Step 4	address-family { <i>ipv4</i> <i>ipv6</i> } Example: RP/0/RP0/CPU0:router(config-eigrp-vrf)# address-family ipv4	Enters a VRF address family configuration mode.
Step 5	router-id <i>router-id</i> Example: RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# router-id 33	Configures a router ID for the EIGRP process.
Step 6	autonomous-system <i>as-number</i> Example: RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# autonomous-system 2	Configures an EIGRP routing process to run within the VRF instance. Note You must configure the autonomous system under VRF configuration to bring-up the VRF interface.
Step 7	redistribute {{ <i>bgp</i> <i>connected</i> <i>isis</i> <i>ospf</i> <i>ospfv3</i> <i>rip</i> <i>static</i> } [<i>as-number</i> <i>instance-name</i>]} [route-policy <i>name</i>] Example: RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# redistribute bgp 100	Injects routes from one routing domain into EIGRP.
Step 8	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# interface gigabitEthernet 0/1/5/0	Configures the interface on which EIGRP the routing protocol runs.
Step 9	site-of-origin { <i>as-number: number</i> <i>ip-address : number</i> } Example: RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)# site-of-origin 3:4	Configures the site-of-origin (SoO) filtering on the EIGRP interface.
Step 10	commit	

Redistributing BGP Routes into EIGRP

Perform this task to redistribute BGP routes into EIGRP.

Typically, EIGRP routes are redistributed into BGP with extended community information appended to the route. BGP carries the route over the VPN backbone with the EIGRP-specific information encoded in the BGP extended community attributes. After the peering customer site receives the route, EIGRP redistributes

the BGP route then extracts the BGP extended community information and reconstructs the route as it appeared in the original customer site.

When redistributing BGP routes into EIGRP, the receiving provider edge (PE) EIGRP router looks for BGP extended community information. If the information is received, it is used to recreate the original EIGRP route. If the information is missing, EIGRP uses the configured default metric value.

If the metric values are not derived from the BGP extended community and a default metric is not configured, the route is not advertised to the customer edge (CE) router by the PE EIGRP. When BGP is redistributed into EIGRP, metrics may not be added to the BGP prefix as extended communities; for example, if EIGRP is not running on the other router. In this case, EIGRP is redistributed into BGP with a “no-metrics” option.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **vrf** *vrf-name*
4. **address-family** { **ipv4** | **ipv6** }
5. **redistribute** {{ **bgp** | **connected** | **isis** | **ospf** | **ospfv3** | **rip** | **static** } [*as-number* | *instance-name*] [**route-policy** *name*]
6. **route-policy** *route-policy-name* { **in** | **out** }
7. **default-metric** *bandwidth delay reliability loading mtu*
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router eigrp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router eigrp 100	Specifies the autonomous system number of the routing process to configure an EIGRP routing process.
Step 3	vrf <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config-eigrp)# router eigrp 100	Configures a VRF instance.
Step 4	address-family { ipv4 ipv6 } Example: RP/0/RP0/CPU0:router(config-eigrp-vrf)# address-family ipv4	Enters a VRF address family configuration mode.
Step 5	redistribute {{ bgp connected isis ospf ospfv3 rip static } [<i>as-number</i> <i>instance-name</i>] [route-policy <i>name</i>] Example:	Injects routes from one routing domain into EIGRP.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# redistribute bgp 100	
Step 6	route-policy <i>route-policy-name</i> { in out } Example: RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# route-policy policy_A in	Applies a routing policy to updates advertised to or received from an EIGRP neighbor.
Step 7	default-metric <i>bandwidth delay reliability loading mtu</i> Example: RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# default-metric 1000 100 250 100 1500	Configures metrics for EIGRP.
Step 8	commit	

Monitoring EIGRP Routing

The commands in this section are used to log neighbor adjacency changes, monitor the stability of the routing system, and help detect problems.

SUMMARY STEPS

- configure**
- router eigrp** *as-number*
- address-family** [**ipv4** | **ipv6**]
- log-neighbor-changes**
- log-neighbor-warnings**
- commit**
- clear eigrp** [*as-number*] [**vrf** { *vrf* | **all** }] [**ipv4** | **ipv6**] **neighbors** [*ip-address* | *type interface-path-id*]
- clear eigrp** [*as-number*] [**vrf** { *vrf* | **all** }] [**ipv4** | **ipv6**] **topology** [*prefix mask*] [*prefix / length*]
- show eigrp** [*as-number*] [**vrf** { *vrf* | **all** }] [**ipv4** | **ipv6**] **accounting**
- show eigrp** [*as-number*] [**vrf** { *vrf* | **all** }] [**ipv4** | **ipv6**] **interfaces** [*type interface-path-id*] [**detail**]
- show eigrp** [*as-number*] [**vrf** { *vrf* | **all** }] [**ipv4** | **ipv6**] **neighbors** [**detail**] [*type interface-path-id* | **static**]
- show protocols eigrp** [**vrf** *vrf-name*]
- show eigrp** [*as-number*] [**vrf** { *vrf* | **all** }] [**ipv4** | **ipv6**] **topology** [*ip-address mask*] [**active** | **all-links** | **detail-links** | **pending** | **summary** | **zero-successors**]
- show eigrp** [*as-number*] [**vrf** { *vrf* | **all** }] [**ipv4** | **ipv6**] **traffic**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router eigrp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router eigrp 100	Specifies the autonomous system number of the routing process to configure an EIGRP routing process.
Step 3	address-family [ipv4 ipv6] Example: RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4	Enters an address family configuration mode.
Step 4	log-neighbor-changes Example: RP/0/RP0/CPU0:router(config-eigrp-af)# log-neighbor-changes	Enables the logging of changes in EIGRP neighbor adjacencies.
Step 5	log-neighbor-warnings Example: RP/0/RP0/CPU0:router(config-eigrp-af)# log-neighbor-warnings	Enables the logging of EIGRP neighbor warning messages.
Step 6	commit	
Step 7	clear eigrp [<i>as-number</i>] [vrf { <i>vrf</i> all }] [ipv4 ipv6] neighbors [<i>ip-address</i> <i>type interface-path-id</i>] Example: RP/0/RP0/CPU0:router# clear eigrp 20 neighbors 0/1/0/0	Deletes EIGRP and VPN neighbor entries from the appropriate table.
Step 8	clear eigrp [<i>as-number</i>] [vrf { <i>vrf</i> all }] [ipv4 ipv6] topology [<i>prefix mask</i>] [<i>prefix / length</i>] Example: RP/0/RP0/CPU0:router# clear eigrp topology	Deletes EIGRP and VRF topology entries from the appropriate tab.
Step 9	show eigrp [<i>as-number</i>] [vrf { <i>vrf</i> all }] [ipv4 ipv6] accounting Example: RP/0/RP0/CPU0:router# show eigrp vrf all accounting	Displays prefix accounting information for EIGRP processes.

	Command or Action	Purpose
Step 10	show eigrp [<i>as-number</i>] [vrf { <i>vrf</i> all }] [ipv4 ipv6] interfaces [<i>type interface-path-id</i>] [detail] Example: RP/0/RP0/CPU0:router# show eigrp interfaces detail	Displays information about interfaces configured for EIGRP.
Step 11	show eigrp [<i>as-number</i>] [vrf { <i>vrf</i> all }] [ipv4 ipv6] neighbors [detail] [<i>type interface-path-id</i> static] Example: RP/0/RP0/CPU0:router# show eigrp neighbors 20 detail static	Displays the neighbors discovered by EIGRP.
Step 12	show protocols eigrp [vrf <i>vrf-name</i>] Example: RP/0/RP0/CPU0:router# show protocols eigrp	Displays information about the EIGRP process configuration.
Step 13	show eigrp [<i>as-number</i>] [vrf { <i>vrf</i> all }] [ipv4 ipv6] topology [<i>ip-address mask</i>] [active all-links detail-links pending summary zero-successors] Example: RP/0/RP0/CPU0:router# show eigrp topology 10.0.0.1 253.254.255.255 summary	Displays entries in the EIGRP topology table.
Step 14	show eigrp [<i>as-number</i>] [vrf { <i>vrf</i> all }] [ipv4 ipv6] traffic Example: RP/0/RP0/CPU0:router# show eigrp traffic	Displays the number of EIGRP packets sent and received.

Configuring an EIGRP Authentication Keychain

Perform the following tasks to configure an authentication keychain on EIGRP interfaces.

Configuring an Authentication Keychain for an IPv4/IPv6 Interface on a Default VRF

Perform this task to configure an authentication keychain for an IPv4/IPv6 interface on a default VRF.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **address-family** { **ipv4** | **ipv6** }

4. **interface** *type interface-path-id*
5. **authentication keychain** *keychain-name*
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router eigrp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router eigrp 100	Specifies the autonomous system number of the routing process to configure an EIGRP routing process.
Step 3	address-family { ipv4 ipv6 } Example: RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4	Enters a VRF address family configuration mode.
Step 4	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-eigrp-af)# interface gigabitEthernet 0/1/5/0	Configures the interface on which EIGRP the routing protocol runs.
Step 5	authentication keychain <i>keychain-name</i> Example: RP/0/RP0/CPU0:router(config-eigrp-af-if)# authentication keychain	Authenticates all EIGRP protocol traffic on the interface, based on the MD5 algorithm.
Step 6	commit	

Configuring an Authentication Keychain for an IPv4/IPv6 Interface on a Nondefault VRF

Perform this task to configure an authentication keychain for an IPv4/IPv6 interface on a nondefault VRF.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **vrf** *vrf-name*
4. **address-family** { **ipv4** | **ipv6** }
5. **interface** *type interface-path-id*
6. **authentication keychain** *keychain-name*
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router eigrp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router eigrp 100	Specifies the autonomous system number of the routing process to configure an EIGRP routing process.
Step 3	vrf <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config-eigrp)# vrf vrf1	Creates a VRF instance and enters VRF configuration mode.
Step 4	address-family { <i>ipv4</i> <i>ipv6</i> } Example: RP/0/RP0/CPU0:router(config-eigrp-vrf)# address-family ipv4	Enters a VRF address family configuration mode.
Step 5	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# interface gigabitEthernet 0/1/5/0	Configures the interface on which EIGRP runs.
Step 6	authentication keychain <i>keychain-name</i> Example: RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)# authentication keychain	Authenticates all EIGRP protocol traffic on the interface, based on the MD5 algorithm.
Step 7	commit	

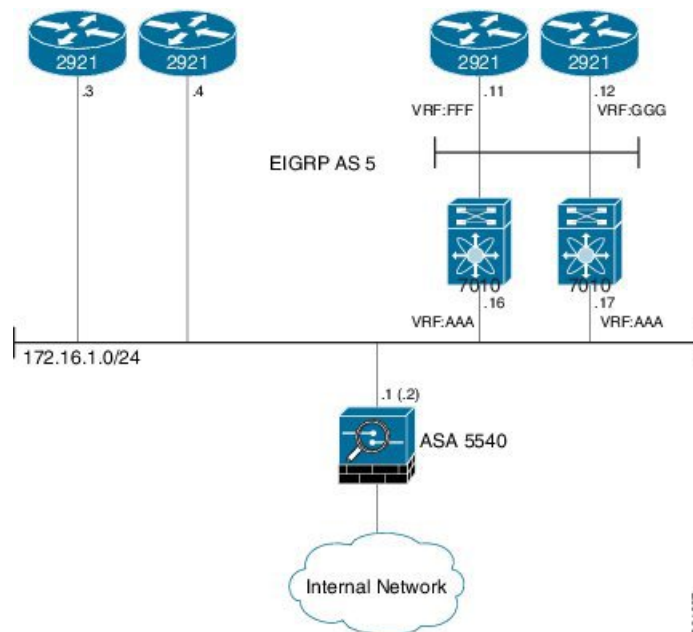
Configuring unicast neighbors

EIGRP typically broadcasts or multicasts routing updates. For security reasons, you can opt to configure static neighbors in the EIGRP routing process, forcing EIGRP to communicate to specified neighbors using unicast. When you specify a static neighbor relationship over a particular interface, EIGRP disables the processing of multicast EIGRP packets on the specified interface. This ensures that EIGRP does not send nor process received multicast EIGRP traffic on an interface which has a static neighbor defined under the EIGRP routing process.

In cases where the neighbors are not adjacent, normal EIGRP peering mechanisms cannot be used to exchange EIGRP information. In order to support this type of network, EIGRP provides the neighbor command, which allows remote neighbors to be configured and sessions established through unicast packet transmission. However, as the number of forwarders needing to exchange EIGRP information over the networking cloud increases, unicast EIGRP neighbor definitions may become cumbersome to manage. Each neighbor must be manually configured, resulting in increased operational costs. To better accommodate deployment of these

topologies, ease configuration management, and reduce operational costs, the Dynamic Neighbors feature provides support for the dynamic discovery of remote unicast (referred to as “remote neighbors”). Remote neighbor support allows EIGRP peering to one or more remote neighbors, which may not be known at the time the device is configured, thus reducing configuration management.

In the topology illustrated below, ASA behaves as a hub and the other routers (2921s, 7010s) act as spokes. The 2921's and 7010's must not peer with each other, and there must never be a time where a packet (data traffic) is routed in this path: ASA > 2921.3 > 2921.4. To support this type of network, EIGRP allows you to configure static neighbors and establish sessions using unicast packet transmission. Thus, in this topology, 2921s and 7010s peer with ASA using neighbor command and ASA is configured to dynamically discover remote neighbors.



Remote Neighbor Session Policy

When using remote unicast-listen or remote multicast-group neighbor configurations, EIGRP neighbor IP addresses are not predefined, and neighbors may be many hops away. A device with this configuration could peer with any device that sends a valid HELLO packet. Because of security considerations, this open aspect requires policy capabilities to limit peering to valid devices and to restrict the number of neighbors in order to limit resource consumption. This capability is accomplished using the following manually configured parameters, and takes effect immediately.

• Neighbor Filter List

The optional allow-list keyword, available in the remote-neighbors command, enables you to use an access list (access control list) to specify the remote IP addresses from which EIGRP neighbor connections may be accepted. If you do not use the allow-list keyword, then all IP addresses (permit any) will be accepted. The access control list (ACL) defines a range of IPv4 or IPv6 IP addresses with the following conditions:

- Any neighbor that has a source IP address that matches an IP address in the access list will be allowed (or denied) based on the user configuration.
- If the allow-list keyword is not specified, any IP address will be permitted (permit any).

- The allow-list keyword is supported only for remote multicast-group and unicast-listen neighbors. It is not available for static, remote static, or local neighbors.
- Incoming EIGRP packets that do not match the specified access list will be rejected.

- **Maximum Remote Neighbors**

The optional max-neighbors keyword, available in the remote-neighbors command, enables you to specify a maximum number of remote neighbors that EIGRP can create using the remote neighbor configurations. When the maximum number of remote neighbors has been created for a configuration, EIGRP rejects all subsequent connection attempts for that configuration. This option helps to protect against denial-of-service attacks that attempt to create many remote neighbors in an attempt to overwhelm device resources. The max-neighbors configuration option has the following conditions:

- This option is supported only for remote multicast-group or unicast-listen neighbors. It is not available for local, static, or remote static neighbors.
- There is no default maximum. If you do not specify a maximum number of remote neighbors, the number of remote neighbors is limited only by available memory and bandwidth.
- Reducing the maximum number of remote neighbors to less than the current number of sessions will result in the neighbors (in no specific order) being dropped until the count reaches the new limit.

- **Configuration Changes for the Neighbor Filter List and Maximum Number of Remote Neighbors**

When the allow-list or max-neighbors configurations are changed, any existing remote EIGRP sessions that are no longer allowed by the new configuration will be removed automatically and immediately. Pre-existing neighbors that are still allowed by the new configuration will not be affected.

Understanding Neighbor Terms

The following terms are used when describing neighbor types:

- **local neighbor:** A neighbor that is adjacent on a shared subnet (or common subnet) and uses a link-local multicast address for packet exchange. This is the default type of neighbor in EIGRP.
- **static Neighbor:** Any neighbor that uses unicast to communicate, is one hop away, is on a common subnet, and whose IP address has been specified using the neighbor ip-address command.
- **remote neighbor:** Any neighbor that is multiple hops away, including Remote Static Neighbors.
- **remote group:** Any neighbor that is multiple hops away, does not have its address manually configured with the neighbor command and uses the multicast group address for packet exchange.
- **remote static neighbor:** Any neighbor that uses unicast to communicate, is multiple hops away, and whose IP address has been specified using the neighbor ip-address command.
- **remote unicast-listen (or simply unicast-listen):** Any neighbor that uses unicast to communicate, is multiple hops away, and whose IP address has not been configured using the neighbor ip-address command.
- **remote dynamic:** Any neighbor that is multiple hops away and whose address has not been configured with the neighbor ip-address command, that is, a remote multicast-group or remote unicast-listen neighbor, but not a remote static neighbor.

Remote Unicast-Listen (Point-to-Point) Neighbors

For configurations in which multiple remote neighbors peer with a single hub (point-to-point), the hub can be configured for remote unicast-listen peering using the remote-neighbors command to allow the remote neighbors to peer with the hub without having to manually configure the remote neighbor IP addresses on the hub. When configured with this command, the hub device:

- Uses its interface IP address as the source IP address for any unicast transmissions. This IP address must be routable.
- Requires neighbors that peer with the hub to be configured using the neighbor ip-address loopback loopback-interface-number remote maximum-hops command where ip-address is the unicast address of the local device interface.
- Listens for unicast HELLO packets on the interface specified in the remote-neighbor command.
- Accepts a unicast HELLO packet if it is in the IP address range configured using the allow-list keyword, or any unicast HELLO packet if an allow list is not defined.
- Rejects multicast HELLO packets from any neighbor that is also sending unicast HELLO packets and is permitted by the unicast allow list (or all neighbors if an allow list is not defined).
- Begins normal neighbor establishment using the IP addresses of the remote neighbors for packet transmission once the neighbor relationship is established.
- When remote-neighbor command is configured on an interface, the router will only start sending HELLOs on that interface if it has at least one neighbor and only to those neighbors from which it has received HELLOs.
- On an interface if dynamic neighbors already exist and remote-neighbor unicast- listen is configured, then the existing neighbor relationships will be torn down and only unicast-neighbor relationships will be allowed there after.

Restrictions for remote neighbors

A single unicast address can only be configured to a single remote static neighbor for a given address-family. You cannot configure a second remote static neighbor using the same unicast address, on a different interface. EIGRP configuration of remote neighbors under different address families is unrestricted.

A single interface can be configured under a single address family with a single unicast-listen remote-neighbors command and with any number of static and remote static neighbors (each using a different unicast address).

Inheritance and precedence of the remote neighbor configurations

Static neighbors configured with the neighbor <address> or neighbor <address> remote commands take precedence over the remote neighbors created as a result of the remote-neighbors command. If the remote address of an incoming unicast EIGRP connection matches both a static neighbor and the remote unicast-listen neighbor access list, the static neighbor is used and no remote unicast-listen neighbor is created. If you configure a new static neighbor while a remote neighbor for the same remote address already exists, EIGRP automatically removes the remote unicast-listen neighbor.

How to configure remote unicast neighbors

When configuring an EIGRP unicast neighbor, the neighbor statement is required on both ends (hub and spoke) of the neighbor relationship in the EIGRP routing process that operate in the same autonomous system.

Before you begin

Ensure that when using unicast-listen mode, IP connectivity (reachability) exists between devices that need to do remote peering.

SUMMARY STEPS

1. **configure**
2. **router eigrp AS Number**
3. **address-family { ipv4 | ipv6 }**

4. `interface-type interface-path-id`
5. `remote-neighbor unicast-listen` {[`allow-list`route policy name][`max-neighbors`maximum remote peers]}
6. `commit`
7. `sh run router eigrp`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	<code>router eigrp AS Number</code> Example: RP/0/RSP0/CPU0:HUB(config)#router eigrp 100	Enables an EIGRP router instance.
Step 3	<code>address-family { ipv4 ipv6 }</code> Example: RP/0/RSP0/CPU0:HUB(config-eigrp)#address-family ipv6	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.
Step 4	<code>interface-type interface-path-id</code> Example: RP/0/RSP0/CPU0:HUB(config-eigrp-af)#int g0/0/0/3	Configures an interface and enters the interface configuration mode.
Step 5	<code>remote-neighbor unicast-listen</code> {[<code>allow-list</code> route policy name][<code>max-neighbors</code> maximum remote peers]} Example: RP/0/RSP0/CPU0:HUB(config-eigrp-af-if)#remote-neighbor unicast-listen	Configures a EGRP process that enables remote neighbors to accept inbound connections from any remote IP address. <ul style="list-style-type: none"> • allow-list keyword to use an access list (access control list) to specify the remote IP addresses from which EIGRP neighbor connections may be accepted. If you do not use the allow list keyword, all IP addresses will be accepted. • max-neighbors keyword to specify the maximum number of remote neighbors. If you do not specify a number, the maximum number of remote neighbors is limited only by available memory and bandwidth.
Step 6	<code>commit</code>	
Step 7	<code>sh run router eigrp</code>	

Configuring EIGRP remote unicast neighbors

The following examples show how to configure both devices (hub and spoke) involved in the neighbor relationship.

```
RP/0/RSP0/CPU0:HUB(config)#router eigrp 100
RP/0/RSP0/CPU0:HUB(config-eigrp)#address-family ipv4
RP/0/RSP0/CPU0:HUB(config-eigrp-af)#int g0/0/0/3
RP/0/RSP0/CPU0:HUB(config-eigrp-af-if)#exit
RP/0/RSP0/CPU0:HUB(config-eigrp-af)#interface gigabitEthernet 0/0/0/3
RP/0/RSP0/CPU0:HUB(config-eigrp-af-if)#remote-neighbor unicast-listen
RP/0/RSP0/CPU0:HUB(config-eigrp-af-if)#commit
```

```
RP/0/RSP0/CPU0:spoke(config)#router eigrp 100
RP/0/RSP0/CPU0:spoke(config-eigrp)#address-family ipv4
RP/0/RSP0/CPU0:spoke(config-eigrp-af)#interface g0/0/0/3
RP/0/RSP0/CPU0:spoke(config-eigrp-af-if)#neighbor 21.21.21.1 remote 10
RP/0/RSP0/CPU0:spoke(config-eigrp-af-if)#commit

RP/0/RSP0/CPU0:spoke#sh run router eigrp
Fri Aug  8 08:47:48.556 UTC
router eigrp 100
address-family ipv4
interface GigabitEthernet0/0/0/3
neighbor 21.21.21.1 remote 10! !!
```

Configuration Examples for Implementing EIGRP

This section provides the following configuration examples:

Configuring a Basic EIGRP Configuration: Example

The following example shows how to configure EIGRP with a policy that filters incoming routes. This is a typical configuration for a router that has just one neighbor, but advertises other connected subnets.

```
router eigrp 144
address-family ipv4
metric maximum-hops 20
router-id 10.10.9.4
route-policy GLOBAL_FILTER_POLICY in
log-neighbor-changes
log-neighbor-warnings
interface Loopback0
!
interface GigabitEthernet 0/2/0/0
passive-interface
!
interface GigabitEthernet 0/6/0/0
hello-interval 8
hold-time 30
summary-address 10.0.0.0 255.255.0.0
!
```

Configuring an EIGRP Stub Operation: Example

The following example shows how to configure an EIGRP stub. Stub operation allows only connected, static, and summary routes to be advertised to neighbors.

```
router eigrp 200
address-family ipv4
stub connected static summary
router-id 172.16.82.22
log-neighbor-changes
log-neighbor-warnings
redistribute connected route-policy CONN_POLICY
interface GigabitEthernet0/6/0/0
passive-interface
```

```

        neighbor 10.0.0.31
    !
    interface GigabitEthernet0/6/0/1
        passive-interface
        neighbor 10.0.1.21
    !
    !
    !

```

Configuring an EIGRP PE-CE Configuration with Prefix-Limits: Example

The following example shows how to configure EIGRP to operate as a PE-CE protocol on a PE router. The configuration is under VRF CUSTOMER_1. A maximum prefix is typically configured to ensure that one set of customer routes do not overwhelm the EIGRP process.

```

router eigrp 500
  vrf CUSTOMER_1
    address-family ipv4
      timers nsf route-hold 300
      router-id 172.16.6.11
      maximum-prefix 450 70
      default-metric 200000 10000 195 10 1500
      log-neighbor-changes
      log-neighbor-warnings
      redistribute maximum-prefix 350 70
      redistribute bgp 1.65500 route-policy SITE_1_POLICY
      interface GigabitEthernet 0/4/0/5
      neighbor 10.22.1.1
    !
  !

```

Configuring an EIGRP Authentication Keychain: Example

The following example shows how to configure an authentication keychain for an IPv4 interface on a nondefault VRF:

```

config
router eigrp 100
  vrf vrf1
  address-family ipv4
  interface POS 0/1/0/0
  authentication keychain key1

```

The following example shows how to configure an authentication keychain for an IPv6 interface on a default VRF:

```

config
router eigrp 100
  address-family ipv6
  interface POS 0/1/0/0
  authentication keychain key2

```

Additional References

The following sections provide references related to implementing EIGRP.

Related Documents

Related Topic	Document Title
EIGRP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Routing Command Reference for Cisco CRS Routers</i>
MPLS VPN support for EIGRP feature information	<i>Implementing MPLS Layer 3 VPNs module and Implementing MPLS Layer 2 VPNs module in MPLS Configuration Guide for the Cisco CRS Routers</i>
Site of Origin (SoO) support for EIGRP feature information	<i>Implementing MPLS Traffic Engineering on Cisco IOS XR Software module in MPLS Configuration Guide for the Cisco CRS Routers</i>
MIB Reference	<i>Cisco Carrier Routing System and Cisco XR 12000 Series Router MIB Support Guide.</i>

Standards

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

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

RFCs

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

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport



CHAPTER 6

Implementing IS-IS

Integrated Intermediate System-to-Intermediate System (IS-IS), Internet Protocol Version 4 (IPv4), is a standards-based Interior Gateway Protocol (IGP). The Cisco software implements the IP routing capabilities described in International Organization for Standardization (ISO)/International Engineering Consortium (IEC) 10589 and RFC 1195, and adds the standard extensions for single topology and multitopology IS-IS for IP Version 6 (IPv6).

This module describes how to implement IS-IS (IPv4 and IPv6) on your Cisco IOS XR network.

- [Prerequisites for Implementing IS-IS, on page 347](#)
- [Implementing IS-IS, on page 347](#)
- [Configuration Examples for Implementing IS-IS , on page 348](#)
- [Where to Go Next, on page 350](#)
- [Additional References, on page 350](#)

Prerequisites for Implementing IS-IS

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.

Implementing IS-IS

Multiple IS-IS instances can exist on the same physical interface. However, you must configure different instance-id for every instance that shares the same physical interface.

Alternatively, you can also create dot1q sub-interfaces and configure each dot1q sub-interface to different IS-IS instances.



Note Users can configure the **no max-metric** command only with levels 1 or 2, that is, **no max-metric level {1|2}** in order to view the result in the output of the **show configuration** command. Else, the maximum metric configuration is not displayed in the output. This behavior is observed before committing the configuration to the router.

Configuration Examples for Implementing IS-IS

This section provides the following configuration examples:

Configuring Single-Topology IS-IS for IPv6: Example

The following example shows single-topology mode being enabled. An IS-IS instance is created, the NET is defined, IPv6 is configured along with IPv4 on an interface, and IPv4 link topology is used for IPv6.

This configuration allows POS interface 0/3/0/0 to form adjacencies for both IPv4 and IPv6 addresses.

```
router isis isp
 net 49.0000.0000.0001.00
 address-family ipv6 unicast
  single-topology
 interface POS0/3/0/0
  address-family ipv4 unicast
  !
  address-family ipv6 unicast
  !
  exit
 !
 interface POS0/3/0/0
  ipv4 address 10.0.1.3 255.255.255.0
  ipv6 address 2001::1/64
```

Configuring Multitopology IS-IS for IPv6: Example

The following example shows multitopology IS-IS being configured in IPv6.

```
router isis isp
 net 49.0000.0000.0001.00
 interface POS0/3/0/0
  address-family ipv6 unicast
  metric-style wide level 1
  exit
 !
 interface POS0/3/0/0
  ipv6 address 2001::1/64
```

Redistributing IS-IS Routes Between Multiple Instances: Example

The following example shows usage of the **attached-bit send always-set** and **redistribute** commands. Two instances, instance “1” restricted to Level 1 and instance “2” restricted to Level 2, are configured.

The Level 1 instance is propagating routes to the Level 2 instance using redistribution. Note that the administrative distance is explicitly configured higher on the Level 2 instance to ensure that Level 1 routes are preferred.

Attached bit is being set for the Level 1 instance since it is redistributing routes into the Level 2 instance. Therefore, instance “1” is a suitable candidate to get from the area to the backbone.

```

router isis 1
  is-type level-2-only
  net 49.0001.0001.0001.0001.00
  address-family ipv4 unicast
  distance 116
  redistribute isis 2 level 2
!
interface GigabitEthernet 0/3/0/0
  address-family ipv4 unicast
!
!
router isis 2
  is-type level-1
  net 49.0002.0001.0001.0002.00
  address-family ipv4 unicast

attached-
bit send always-
set
!
interface GigabitEthernet 0/1/0/0
  address-family ipv4 unicast

```

Tagging Routes: Example

The following example shows how to tag routes.

```

route-policy isis-tag-55
end-policy
!
route-policy isis-tag-555
  if destination in (5.5.5.0/24 eq 24) then
    set tag 555
    pass
  else
    drop
  endif
end-policy
!
router static
  address-family ipv4 unicast
  0.0.0.0/0 2.6.0.1
  5.5.5.0/24 Null0
!
!
router isis uut
  net 00.0000.0000.12a5.00
  address-family ipv4 unicast
  metric-style wide
  redistribute static level-1 route-policy isis-tag-555
  spf prefix-priority critical tag 13
  spf prefix-priority high tag 444
  spf prefix-priority medium tag 777

```

Configuring IS-IS Overload Bit Avoidance: Example

The following example shows how to activate IS-IS overload bit avoidance:

```
config
mpls traffic-eng path-selection ignore overload
```

The following example shows how to deactivate IS-IS overload bit avoidance:

```
config
no mpls traffic-eng path-selection ignore overload
```

Where to Go Next

To implement more IP routing protocols, see the following document modules in *Routing Configuration Guide for Cisco CRS Routers*:

- Implementing OSPF
- Implementing BGP
- Implementing EIGRP
- Implementing RIP

Additional References

The following sections provide references related to implementing IS-IS.

Related Documents

Related Topic	Document Title
IS-IS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Routing Command Reference for Cisco CRS Routers</i>
MPLS TE feature information	<i>Implementing MPLS Traffic Engineering on Cisco IOS XR Software</i> module in <i>MPLS Configuration Guide for the Cisco CRS Routers</i>
Bidirectional Forwarding Detection (BFD)	<i>Interface and Hardware Component Configuration Guide for Cisco CRS Routers</i> and <i>Interface and Hardware Component Command Reference for Cisco CRS Routers</i>

Standards

Standards	Title
Draft-ietf-isis-ipv6-05.txt	<i>Routing IPv6 with IS-IS</i> , by Christian E. Hopps
Draft-ietf-isis-wg-multi-topology-06.txt	<i>M-ISIS: Multi Topology (MT) Routing in IS-IS</i> , by Tony Przygienda, Naiming Shen, and Nischal Sheth
Draft-ietf-isis-traffic-05.txt	<i>IS-IS Extensions for Traffic Engineering</i> , by Henk Smit and Toni Li
Draft-ietf-isis-restart-04.txt	<i>Restart Signaling for IS-IS</i> , by M. Shand and Les Ginsberg
Draft-ietf-isis-igp-p2p-over-lan-05.txt	<i>Point-to-point operation over LAN in link-state routing protocols</i> , by Naiming Shen
Draft-ietf-rtgwg-ipfir-framework-06.txt	<i>IP Fast Reroute Framework</i> , by M. Shand and S. Bryant
Draft-ietf-rtgwg-lf-conv-fmwk-00.txt	<i>A Framework for Loop-free Convergence</i> , by M. Shand and S. Bryant

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

RFCs

RFCs	Title
RFC 1142	OSI IS-IS Intra-domain Routing Protocol
RFC 1195	Use of OSI IS-IS for Routing in TCP/IP and Dual Environments
RFC 2763	Dynamic Hostname Exchange Mechanism for IS-IS
RFC 2966	Domain-wide Prefix Distribution with Two-Level IS-IS
RFC 2973	IS-IS Mesh Groups
RFC 3277	IS-IS Transient Blackhole Avoidance

RFCs	Title
RFC 3373	Three-Way Handshake for IS-IS Point-to-Point Adjacencies
RFC 3567	IS-IS Cryptographic Authentication
RFC 4444	IS-IS Management Information Base

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport



CHAPTER 7

Implementing OSPF

Open Shortest Path First (OSPF) is an Interior Gateway Protocol (IGP) developed by the OSPF working group of the Internet Engineering Task Force (IETF). Designed expressly for IP networks, OSPF supports IP subnetting and tagging of externally derived routing information. OSPF also allows packet authentication and uses IP multicast when sending and receiving packets.

OSPF Version 3 (OSPFv3) expands on OSPF Version 2, providing support for IPv6 routing prefixes.

This module describes the concepts and tasks you need to implement both versions of OSPF on your Cisco CRS Router. The term “OSPF” implies both versions of the routing protocol, unless otherwise noted.



Note For more information about OSPF on Cisco IOS XR software and complete descriptions of the OSPF commands listed in this module, see the [Related Documents, on page 444](#) section of this module. To locate documentation for other commands that might appear during execution of a configuration task, search online in the *Cisco IOS XR Commands Master List for the Cisco CRS Router*

Feature History for Implementing OSPF

Release	Modification
Release 2.0	This feature was introduced.
Release 3.2.2	Support was added for OSPFv3 Graceful Restart.
Release 3.3.0	Support was added for the following features: <ul style="list-style-type: none">• Multicast-Intact for OSPFv2• Interface Association to a VRF• OSPF Provider Edge to Customer Edge (PE-CE) Protocol• Multiple OSPF Instances (OSPF Process and a VRF)• RPL-based Type 3 Filtering• LSA Pacing

Release	Modification
Release 3.4.0	Support was added for the following features: <ul style="list-style-type: none"> • OSPF Forwarding Adjacency • OSPF SNMP Trap MIB
Release 3.4.1	Support was added for the multi-area adjacency feature.
Release 3.5.0	Support was added for the following features: <ul style="list-style-type: none"> • Label Distribution Protocol IGP Auto-configuration for OSPF • OSPF Authentication Message Digest Management • GTSM TTL Security Mechanism for OSPF • Path Computation Element for OSPFv2 • OSPF Warm Standby
Release 3.6.0	Support was added for the following features: <ul style="list-style-type: none"> • OSPFv2 Sham Link Support for MPLS VPN • OSPFv2 nonstop routing (NSR)
Release 3.7.0	OSPFv2 Sham Link and Nonstop Routing for OSPFv2 were added.
Release 3.8.0	Support was added for the following features: <ul style="list-style-type: none"> • LSA refresh interval configuration • OSPF queue tuning parameters • OSPFv2 scale enhancements to improve event processing and performance in a scaled configuration environment
Release 3.9.0	Support was added for the following features: <ul style="list-style-type: none"> • OSPF Over Generic Routing Encapsulation (GRE) tunnel interfaces • OSPFv2 SPF Prefix Prioritization. • IP fast reroute loop-free alternates computation • Warm Standby and Nonstop Routing for OSPF Version 3
Release 4.2.0	Support was added for the following features: <ul style="list-style-type: none"> • OSPFv2 Fast Re-route Per-Prefix Computation • OSPFv3 Non-stop Routing (NSR)

Release	Modification
Release 4.2.1	Support was added for the following features: <ul style="list-style-type: none"> • OSPFv3 SPF Prefix Prioritization. • Management Information Base (MIB) for OSPFv3
Release 4.3.0	Support was added for the following features: <ul style="list-style-type: none"> • OSPFv2 VRF Lite • OSPFv3 Timers Update
Release 5.3.0	Support was added for OSPFv2 Segment Routing Topology Independent Fast Reroute
Release 5.3.2	Support was added for the following features: <ul style="list-style-type: none"> • Support was added for Segment Routing OSPF and LDP Interworking • OSPF strict-mode Support for BFD Dampening • OSPF FIB Download Notification

- [Prerequisites for Implementing OSPF](#) , on page 355
- [Information About Implementing OSPF](#) , on page 356
- [How to Implement OSPF](#) , on page 383
- [Configuring IP Fast Reroute Loop-free Alternate](#), on page 431
- [Configuration Examples for Implementing OSPF](#) , on page 435
- [Where to Go Next](#), on page 444
- [Additional References](#), on page 444

Prerequisites for Implementing OSPF

The following are prerequisites for implementing OSPF on Cisco IOS XR software:

- 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.
- Configuration tasks for OSPFv3 assume that you are familiar with IPv6 addressing and basic configuration. See the *Implementing Network Stack IPv4 and IPv6 on Cisco IOS XR Software* module of the *IP Addresses and Services Configuration Guide for Cisco CRS Routers* for information on IPv6 routing and addressing.
- Before you enable OSPFv3 on an interface, you must perform the following tasks:
 - Complete the OSPF network strategy and planning for your IPv6 network. For example, you must decide whether multiple areas are required.
 - Enable IPv6 on the interface.

- Configuring authentication (IP Security) is an optional task. If you choose to configure authentication, you must first decide whether to configure plain text or Message Digest 5 (MD5) authentication, and whether the authentication applies to an entire area or specific interfaces.

Information About Implementing OSPF

To implement OSPF you need to understand the following concepts:

OSPF Functional Overview

OSPF is a routing protocol for IP. It is a link-state protocol, as opposed to a distance-vector protocol. A link-state protocol makes its routing decisions based on the states of the links that connect source and destination machines. The state of the link is a description of that interface and its relationship to its neighboring networking devices. The interface information includes the IP address of the interface, network mask, type of network to which it is connected, routers connected to that network, and so on. This information is propagated in various types of link-state advertisements (LSAs).

A router stores the collection of received LSA data in a link-state database. This database includes LSA data for the links of the router. The contents of the database, when subjected to the Dijkstra algorithm, extract data to create an OSPF routing table. The difference between the database and the routing table is that the database contains a complete collection of raw data; the routing table contains a list of shortest paths to known destinations through specific router interface ports.

OSPF is the IGP of choice because it scales to large networks. It uses areas to partition the network into more manageable sizes and to introduce hierarchy in the network. A router is attached to one or more areas in a network. All of the networking devices in an area maintain the same complete database information about the link states in their area only. They do not know about all link states in the network. The agreement of the database information among the routers in the area is called convergence.

At the intradomain level, OSPF can import routes learned using Intermediate System-to-Intermediate System (IS-IS). OSPF routes can also be exported into IS-IS. At the interdomain level, OSPF can import routes learned using Border Gateway Protocol (BGP). OSPF routes can be exported into BGP.

Unlike Routing Information Protocol (RIP), OSPF does not provide periodic routing updates. On becoming neighbors, OSPF routers establish an adjacency by exchanging and synchronizing their databases. After that, only changed routing information is propagated. Every router in an area advertises the costs and states of its links, sending this information in an LSA. This state information is sent to all OSPF neighbors one hop away. All the OSPF neighbors, in turn, send the state information unchanged. This flooding process continues until all devices in the area have the same link-state database.

To determine the best route to a destination, the software sums all of the costs of the links in a route to a destination. After each router has received routing information from the other networking devices, it runs the shortest path first (SPF) algorithm to calculate the best path to each destination network in the database.

The networking devices running OSPF detect topological changes in the network, flood link-state updates to neighbors, and quickly converge on a new view of the topology. Each OSPF router in the network soon has the same topological view again. OSPF allows multiple equal-cost paths to the same destination. Since all link-state information is flooded and used in the SPF calculation, multiple equal cost paths can be computed and used for routing.

On broadcast and nonbroadcast multiaccess (NBMA) networks, the designated router (DR) or backup DR performs the LSA flooding. On point-to-point networks, flooding simply exits an interface directly to a neighbor.

OSPF runs directly on top of IP; it does not use TCP or User Datagram Protocol (UDP). OSPF performs its own error correction by means of checksums in its packet header and LSAs.

In OSPFv3, the fundamental concepts are the same as OSPF Version 2, except that support is added for the increased address size of IPv6. New LSA types are created to carry IPv6 addresses and prefixes, and the protocol runs on an individual link basis rather than on an individual IP-subnet basis.

OSPF typically requires coordination among many internal routers: Area Border Routers (ABRs), which are routers attached to multiple areas, and Autonomous System Border Routers (ASBRs) that export reroutes from other sources (for example, IS-IS, BGP, or static routes) into the OSPF topology. At a minimum, OSPF-based routers or access servers can be configured with all default parameter values, no authentication, and interfaces assigned to areas. If you intend to customize your environment, you must ensure coordinated configurations of all routers.

Key Features Supported in the Cisco IOS XR Software OSPF Implementation

The Cisco IOS XR Software implementation of OSPF conforms to the OSPF Version 2 and OSPF Version 3 specifications detailed in the Internet RFC 2328 and RFC 2740, respectively.

The following key features are supported in the Cisco IOS XR Software implementation:

- Hierarchy—CLI hierarchy is supported.
- Inheritance—CLI inheritance is supported.
- Stub areas—Definition of stub areas is supported.
- NSF—Nonstop forwarding is supported.
- SPF throttling—Shortest path first throttling feature is supported.
- LSA throttling—LSA throttling feature is supported.
- Fast convergence—SPF and LSA throttle timers are set, configuring fast convergence. The OSPF LSA throttling feature provides a dynamic mechanism to slow down LSA updates in OSPF during network instability. LSA throttling also allows faster OSPF convergence by providing LSA rate limiting in milliseconds.
- Route redistribution—Routes learned using any IP routing protocol can be redistributed into any other IP routing protocol.
- Authentication—Plain text and MD5 authentication among neighboring routers within an area is supported.
- Routing interface parameters—Configurable parameters supported include interface output cost, retransmission interval, interface transmit delay, router priority, router “dead” and hello intervals, and authentication key.
- Virtual links—Virtual links are supported.
- Not-so-stubby area (NSSA)—RFC 1587 is supported.
- OSPF over demand circuit—RFC 1793 is supported.

Comparison of Cisco IOS XR Software OSPFv3 and OSPFv2

Much of the OSPFv3 protocol is the same as in OSPFv2. OSPFv3 is described in RFC 2740.

The key differences between the Cisco IOS XR Software OSPFv3 and OSPFv2 protocols are as follows:

- OSPFv3 expands on OSPFv2 to provide support for IPv6 routing prefixes and the larger size of IPv6 addresses.
- When using an NBMA interface in OSPFv3, users must manually configure the router with the list of neighbors. Neighboring routers are identified by the link local address of the attached interface of the neighbor.
- Unlike in OSPFv2, multiple OSPFv3 processes can be run on a link.
- LSAs in OSPFv3 are expressed as “prefix and prefix length” instead of “address and mask.”
- The router ID is a 32-bit number with no relationship to an IPv6 address.

OSPF Hierarchical CLI and CLI Inheritance

Cisco IOS XR Software introduces new OSPF configuration fundamentals consisting of hierarchical CLI and CLI inheritance.

Hierarchical CLI is the grouping of related network component information at defined hierarchical levels such as at the router, area, and interface levels. Hierarchical CLI allows for easier configuration, maintenance, and troubleshooting of OSPF configurations. When configuration commands are displayed together in their hierarchical context, visual inspections are simplified. Hierarchical CLI is intrinsic for CLI inheritance to be supported.

With CLI inheritance support, you need not explicitly configure a parameter for an area or interface. In Cisco IOS XR Software, the parameters of interfaces in the same area can be exclusively configured with a single command, or parameter values can be inherited from a higher hierarchical level—such as from the area configuration level or the router ospf configuration levels.

For example, the hello interval value for an interface is determined by this precedence “IF” statement:

If the **hello interval** command is configured at the interface configuration level, then use the interface configured value, else

If the **hello interval** command is configured at the area configuration level, then use the area configured value, else

If the **hello interval** command is configured at the router ospf configuration level, then use the router ospf configured value, else

Use the default value of the command.

**Tip**

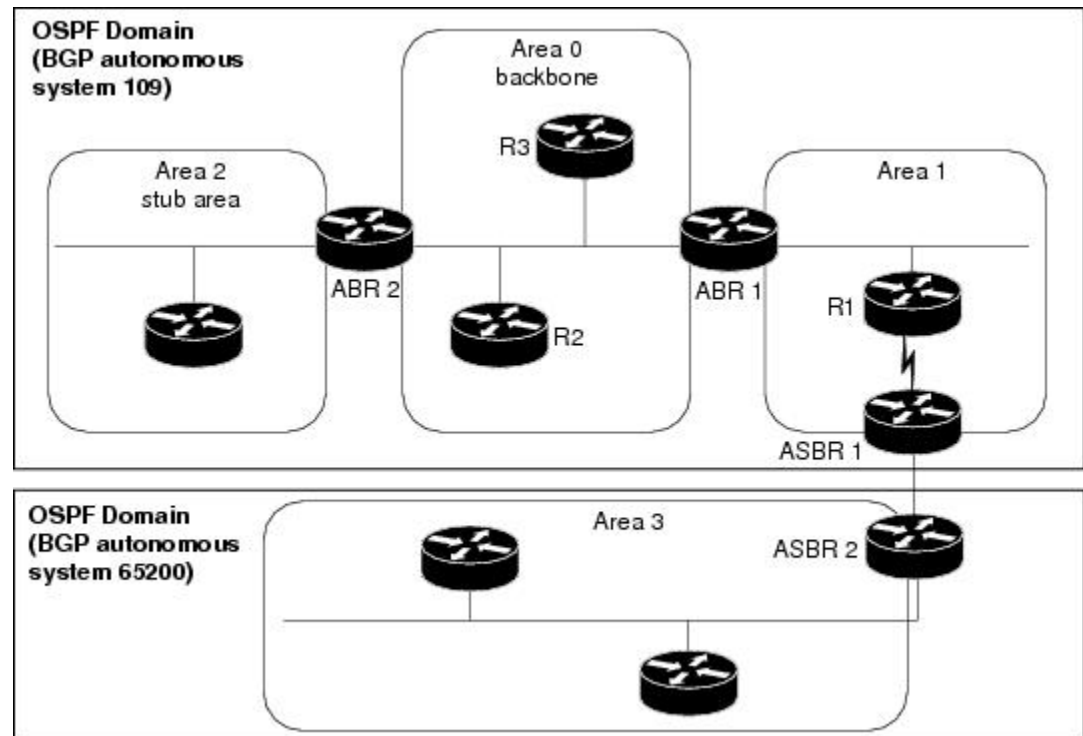
Understanding hierarchical CLI and CLI inheritance saves you considerable configuration time. See [Configuring Authentication at Different Hierarchical Levels for OSPF Version 2, on page 389](#) to understand how to implement these fundamentals. In addition, Cisco IOS XR Software examples are provided in [Configuration Examples for Implementing OSPF, on page 435](#).

OSPF Routing Components

Before implementing OSPF, you must know what the routing components are and what purpose they serve. They consist of the autonomous system, area types, interior routers, ABRs, and ASBRs.

Figure 20: OSPF Routing Components

This figure illustrates the routing components in an OSPF network topology.



Autonomous Systems

The autonomous system is a collection of networks, under the same administrative control, that share routing information with each other. An autonomous system is also referred to as a routing domain. [Figure 20: OSPF Routing Components, on page 359](#) shows two autonomous systems: 109 and 65200. An autonomous system can consist of one or more OSPF areas.

Areas

Areas allow the subdivision of an autonomous system into smaller, more manageable networks or sets of adjacent networks. As shown in [Figure 20: OSPF Routing Components, on page 359](#), autonomous system 109 consists of three areas: Area 0, Area 1, and Area 2.

OSPF hides the topology of an area from the rest of the autonomous system. The network topology for an area is visible only to routers inside that area. When OSPF routing is within an area, it is called *intra-area routing*. This routing limits the amount of link-state information flood into the network, reducing routing traffic. It also reduces the size of the topology information in each router, conserving processing and memory requirements in each router.

Also, the routers within an area cannot see the detailed network topology outside the area. Because of this restricted view of topological information, you can control traffic flow between areas and reduce routing traffic when the entire autonomous system is a single routing domain.

Backbone Area

A backbone area is responsible for distributing routing information between multiple areas of an autonomous system. OSPF routing occurring outside of an area is called *interarea routing*.

The backbone itself has all properties of an area. It consists of ABRs, routers, and networks only on the backbone. As shown in [Figure 20: OSPF Routing Components, on page 359](#), Area 0 is an OSPF backbone area. Any OSPF backbone area has a reserved area ID of 0.0.0.0.

Stub Area

A stub area is an area that does not accept route advertisements or detailed network information external to the area. A stub area typically has only one router that interfaces the area to the rest of the autonomous system. The stub ABR advertises a single default route to external destinations into the stub area. Routers within a stub area use this route for destinations outside the area and the autonomous system. This relationship conserves LSA database space that would otherwise be used to store external LSAs flooded into the area. In [Figure 20: OSPF Routing Components, on page 359](#), Area 2 is a stub area that is reached only through ABR 2. Area 0 cannot be a stub area.

Not-so-Stubby Area

A Not-so-Stubby Area (NSSA) is similar to the stub area. NSSA does not flood Type 5 external LSAs from the core into the area, but can import autonomous system external routes in a limited fashion within the area.

NSSA allows importing of Type 7 autonomous system external routes within an NSSA area by redistribution. These Type 7 LSAs are translated into Type 5 LSAs by NSSA ABRs, which are flooded throughout the whole routing domain. Summarization and filtering are supported during the translation.

Use NSSA to simplify administration if you are a network administrator that must connect a central site using OSPF to a remote site that is using a different routing protocol.

Before NSSA, the connection between the corporate site border router and remote router could not be run as an OSPF stub area because routes for the remote site could not be redistributed into a stub area, and two routing protocols needed to be maintained. A simple protocol like RIP was usually run and handled the redistribution. With NSSA, you can extend OSPF to cover the remote connection by defining the area between the corporate router and remote router as an NSSA. Area 0 cannot be an NSSA.

Routers

The OSPF network is composed of ABRs, ASBRs, and interior routers.

Area Border Routers

An area border routers (ABR) is a router with multiple interfaces that connect directly to networks in two or more areas. An ABR runs a separate copy of the OSPF algorithm and maintains separate routing data for each area that is attached to, including the backbone area. ABRs also send configuration summaries for their attached areas to the backbone area, which then distributes this information to other OSPF areas in the autonomous system. In [Figure 20: OSPF Routing Components, on page 359](#), there are two ABRs. ABR 1 interfaces Area 1 to the backbone area. ABR 2 interfaces the backbone Area 0 to Area 2, a stub area.

Autonomous System Boundary Routers (ASBR)

An autonomous system boundary router (ASBR) provides connectivity from one autonomous system to another system. ASBRs exchange their autonomous system routing information with boundary routers in other autonomous systems. Every router inside an autonomous system knows how to reach the boundary routers for its autonomous system.

ASBRs can import external routing information from other protocols like BGP and redistribute them as AS-external (ASE) Type 5 LSAs to the OSPF network. If the Cisco IOS XR router is an ASBR, you can configure it to advertise VIP addresses for content as autonomous system external routes. In this way, ASBRs flood information about external networks to routers within the OSPF network.

ASBR routes can be advertised as a Type 1 or Type 2 ASE. The difference between Type 1 and Type 2 is how the cost is calculated. For a Type 2 ASE, only the external cost (metric) is considered when multiple paths to the same destination are compared. For a Type 1 ASE, the combination of the external cost and cost to reach the ASBR is used. Type 2 external cost is the default and is always more costly than an OSPF route and used only if no OSPF route exists.

Interior Routers

An interior router (such as R1 in [Figure 20: OSPF Routing Components, on page 359](#)) is attached to one area (for example, all the interfaces reside in the same area).

OSPF Process and Router ID

An OSPF process is a logical routing entity running OSPF in a physical router. This logical routing entity should not be confused with the logical routing feature that allows a system administrator (known as the Cisco IOS XR Software Owner) to partition the physical box into separate routers.

A physical router can run multiple OSPF processes, although the only reason to do so would be to connect two or more OSPF domains. Each process has its own link-state database. The routes in the routing table are calculated from the link-state database. One OSPF process does not share routes with another OSPF process unless the routes are redistributed.

Each OSPF process is identified by a router ID. The router ID must be unique across the entire routing domain. OSPF obtains a router ID from the following sources, in order of decreasing preference:

- By default, when the OSPF process initializes, it checks if there is a router-id in the checkpointing database.
- The 32-bit numeric value specified by the OSPF router-id command in router configuration mode. (This value can be any 32-bit value. It is not restricted to the IPv4 addresses assigned to interfaces on this router, and need not be a routable IPv4 address.)
- The ITAL selected router-id.
- The primary IPv4 address of an interface over which this OSPF process is running. The first interface address in the OSPF interface is selected.

We recommend that the router ID be set by the **router-id** command in router configuration mode. Separate OSPF processes could share the same router ID, in which case they cannot reside in the same OSPF routing domain.

Supported OSPF Network Types

OSPF classifies different media into the following types of networks:

- NBMA networks
- Point-to-point networks (POS)
- Broadcast networks (Gigabit Ethernet)
- Point-to-multipoint

You can configure your Cisco IOS XR network as either a broadcast or an NBMA network. Using this feature, you can configure broadcast networks as NBMA networks when, for example, you have routers in your network that do not support multicast addressing.

Route Authentication Methods for OSPF

OSPF Version 2 supports two types of authentication: plain text authentication and MD5 authentication. By default, no authentication is enabled (referred to as null authentication in RFC 2178).

OSPF Version 3 supports all types of authentication except key rollover.

Plain Text Authentication

Plain text authentication (also known as Type 1 authentication) uses a password that travels on the physical medium and is easily visible to someone that does not have access permission and could use the password to infiltrate a network. Therefore, plain text authentication does not provide security. It might protect against a faulty implementation of OSPF or a misconfigured OSPF interface trying to send erroneous OSPF packets.

MD5 Authentication

MD5 authentication provides a means of security. No password travels on the physical medium. Instead, the router uses MD5 to produce a message digest of the OSPF packet plus the key, which is sent on the physical medium. Using MD5 authentication prevents a router from accepting unauthorized or deliberately malicious routing updates, which could compromise your network security by diverting your traffic.



Note MD5 authentication supports multiple keys, requiring that a key number be associated with a key.

See [OSPF Authentication Message Digest Management, on page 379](#).

Authentication Strategies

Authentication can be specified for an entire process or area, or on an interface or a virtual link. An interface or virtual link can be configured for only one type of authentication, not both. Authentication configured for an interface or virtual link overrides authentication configured for the area or process.

If you intend for all interfaces in an area to use the same type of authentication, you can configure fewer commands if you use the **authentication** command in the area configuration submode (and specify the **message-digest** keyword if you want the entire area to use MD5 authentication and HMAC SHA 256 authentication). This strategy requires fewer commands than specifying authentication for each interface.

Key Rollover

To support the changing of an MD5 key in an operational network without disrupting OSPF adjacencies (and hence the topology), a key rollover mechanism is supported. As a network administrator configures the new key into the multiple networking devices that communicate, some time exists when different devices are using both a new key and an old key. If an interface is configured with a new key, the software sends two copies of the same packet, each authenticated by the old key and new key. The software tracks which devices start using the new key, and the software stops sending duplicate packets after it detects that all of its neighbors are using the new key. The software then discards the old key. The network administrator must then remove the old key from each the configuration file of each router.

Neighbors and Adjacency for OSPF

Routers that share a segment (Layer 2 link between two interfaces) become neighbors on that segment. OSPF uses the hello protocol as a neighbor discovery and keep alive mechanism. The hello protocol involves receiving and periodically sending hello packets out each interface. The hello packets list all known OSPF neighbors on the interface. Routers become neighbors when they see themselves listed in the hello packet of the neighbor. After two routers are neighbors, they may proceed to exchange and synchronize their databases, which creates an adjacency. On broadcast and NBMA networks all neighboring routers have an adjacency.

OSPF strict-mode Support for BFD Dampening

Strict-mode is an OSPF BFD operation mode which keeps the neighbor in a down state until the BFD session is up. The status of the neighbor node shows as awaiting BFD session up in the output of the **show ospf neighbor** command. This will ensure that client protocols do not operate independent of the declared state of BFD.

Restrictions

- Strict-mode and non-strict-mode modes of operation are incompatible and will cause OSPF to never form a neighbor relationship. Strict-mode can not be configured on one node and default/non-strict mode on the other. Both BFD neighbors must run IOS-XR images that support strict-mode. However, if by design the additional BFD clients have already initiated the BFD session and OSPF is not the only initiator, then they may form a neighbor relationship.
- Due to the dependency on BFD, OSPF operating in strict-mode may experience delayed neighbor establishment and full adjacency.

Enabling strict-mode

Strict-mode can be enabled at different levels - process, area and interface. The following procedure describes how to enable BFD strict-mode for Open Shortest Path First (OSPF) on an interface:

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **area** *area-id*
4. **interface** *type interface-path-id*

5. **bfd fast-detect strict-mode**
6. **commit**
7. **show ospf interface type interface-path-id**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: RP/0/RP0/CPU0:router# configure	Enters global configuration mode.
Step 2	router ospf process-name Example: RP/0/RP0/CPU0:router(config)# router ospf 1	Enters OSPF configuration mode, allowing you to configure the OSPF routing process. Use the show ospf command in EXEC configuration mode to obtain the process-name for the current router.
Step 3	area area-id Example: RP/0/RP0/CPU0:router(config-ospf)# area 0	Configures an Open Shortest Path First (OSPF) area. Replace <i>area-id</i> with the OSPF area identifier.
Step 4	interface type interface-path-id Example: RP/0/RP0/CPU0:router(config-ospf-ar)# interface gigabitEthernet 0/3/0/1	Enters interface configuration mode and specifies the interface name and notation <i>rack/slot/module/port</i> . The example indicates a Gigabit Ethernet interface in modular services card slot 3.
Step 5	bfd fast-detect strict-mode Example: RP/0/RP0/CPU0:router(config-ospf-ar-if)# bfd fast-detect strict-mode	Enables strict-mode to hold down neighbor session until BFD session is up.
Step 6	commit	Commits the changes to the running configuration.
Step 7	show ospf interface type interface-path-id Example: RP/0/RP0/CPU0:router(config-ospf-ar-if)# show ospf interface gigabitEthernet 0/3/0/1	Verify that strict-mode is enabled on the appropriate interface.

BFD strict-mode: Example

The following example shows how to enable BFD strict-mode for OSPF on a Gigabit Ethernet interface and check the OSPF interface information. The value of **Mode** displays as **Strict** when BFD strict-mode is enabled. By default, the value of **Mode** displays as **Default**.

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#router ospf 0
RP/0/RP0/CPU0:router(config-ospf)#area 0
RP/0/RP0/CPU0:router(config-ospf-ar)#interface gigabitEthernet 0/3/0/1
```

```
RP/0/RP0/CPU0:router(config-ospf-ar-if)#bfd fast-detect strict-mode
RP/0/RP0/CPU0:router(config-ospf-ar-if)#commit
RP/0/RP0/CPU0:router(config-ospf-ar-if)#end
RP/0/RP0/CPU0:router#show ospf interface gigabitEthernet 0/3/0/1
```

```
GigabitEthernet0/3/0/1 is up, line protocol is up
 Internet Address 10.1.1.2/24, Area 0
  Process ID 1, Router ID 2.2.2.2, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DR, Priority 1, MTU 1500, MaxPktSz 1500
  BFD enabled, BFD interval 150 msec, BFD multiplier 3, Mode: Strict
  Designated Router (ID) 2.2.2.2, Interface address 10.1.1.2
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:07:358
  Index 1/1, flood queue length 0
  Next 0(0)/0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  LS Ack List: current length 0, high water mark 1
  Neighbor Count is 1, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)
  Multi-area interface Count is 0
```

The following example shows the output of the **show ospf neighbor** command. # indicates that the neighbor is waiting for the BFD session to come up.

```
RP/0/RP0/CPU0:router#show ospf neighbor
```

```
Neighbors for OSPF 1
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
1.1.1.1	0	DOWN/DROTHER	00:00:33	10.1.1.3/24	GigabitEthernet0/3/0/1#

```
Total neighbor count: 1
```

OSPF FIB Download Notification

OSPF FIB Download Notification feature minimizes the ingress traffic drop for a prolonged period of time after the line card reloads and this feature is enabled by default.

Open Shortest Path First (OSPF) registers with Routing Information Base (RIB) through Interface Table Attribute Library (ITAL) which keeps the interface down until all the routes are downloaded to Forwarding Information Base (FIB). OSPF gets the Interface Up notification when all the routes on the reloaded line card are downloaded through RIB/FIB.

RIB provides notification to registered clients when a:

- Node is lost.
- Node is created.
- Node's FIB upload is completed.

Designated Router (DR) for OSPF

On point-to-point and point-to-multipoint networks, the Cisco IOS XR software floods routing updates to immediate neighbors. No DR or backup DR (BDR) exists; all routing information is flooded to each router.

On broadcast or NBMA segments only, OSPF minimizes the amount of information being exchanged on a segment by choosing one router to be a DR and one router to be a BDR. Thus, the routers on the segment have a central point of contact for information exchange. Instead of each router exchanging routing updates with every other router on the segment, each router exchanges information with the DR and BDR. The DR and BDR relay the information to the other routers. On broadcast network segments the number of OSPF packets is further reduced by the DR and BDR sending such OSPF updates to a multicast IP address that all OSPF routers on the network segment are listening on.

The software looks at the priority of the routers on the segment to determine which routers are the DR and BDR. The router with the highest priority is elected the DR. If there is a tie, then the router with the higher router ID takes precedence. After the DR is elected, the BDR is elected the same way. A router with a router priority set to zero is ineligible to become the DR or BDR.

Default Route for OSPF

Type 5 (ASE) LSAs are generated and flooded to all areas except stub areas. For the routers in a stub area to be able to route packets to destinations outside the stub area, a default route is injected by the ABR attached to the stub area.

The cost of the default route is 1 (default) or is determined by the value specified in the **default-cost** command.

Link-State Advertisement Types for OSPF Version 2

Each of the following LSA types has a different purpose:

- Router LSA (Type 1)—Describes the links that the router has within a single area, and the cost of each link. These LSAs are flooded within an area only. The LSA indicates if the router can compute paths based on quality of service (QoS), whether it is an ABR or ASBR, and if it is one end of a virtual link. Type 1 LSAs are also used to advertise stub networks.
- Network LSA (Type 2)—Describes the link state and cost information for all routers attached to a multiaccess network segment. This LSA lists all the routers that have interfaces attached to the network segment. It is the job of the designated router of a network segment to generate and track the contents of this LSA.
- Summary LSA for ABRs (Type 3)—Advertises internal networks to routers in other areas (interarea routes). Type 3 LSAs may represent a single network or a set of networks aggregated into one prefix. Only ABRs generate summary LSAs.
- Summary LSA for ASBRs (Type 4)—Advertises an ASBR and the cost to reach it. Routers that are trying to reach an external network use these advertisements to determine the best path to the next hop. ABRs generate Type 4 LSAs.
- Autonomous system external LSA (Type 5)—Redistributes routes from another autonomous system, usually from a different routing protocol into OSPF.
- Autonomous system external LSA (Type 7)—Provides for carrying external route information within an NSSA. Type 7 LSAs may be originated by and advertised throughout an NSSA. NSSAs do not receive

or originate Type 5 LSAs. Type 7 LSAs are advertised only within a single NSSA. They are not flooded into the backbone area or into any other area by border routers.

- Intra-area-prefix LSAs (Type 9)—A router can originate multiple intra-area-prefix LSAs for every router or transit network, each with a unique link-state ID. The link-state ID for each intra-area-prefix LSA describes its association to either the router LSA or network LSA and contains prefixes for stub and transit networks.
- Area local scope (Type 10)—Opaque LSAs are not flooded past the borders of their associated area.
- Link-state (Type 11)—The LSA is flooded throughout the AS. The flooding scope of Type 11 LSAs are equivalent to the flooding scope of AS-external (Type 5) LSAs. Similar to Type 5 LSAs, the LSA is rejected if a Type 11 opaque LSA is received in a stub area from a neighboring router within the stub area. Type 11 opaque LSAs have these attributes:
 - LSAs are flooded throughout all transit areas.
 - LSAs are not flooded into stub areas from the backbone.
 - LSAs are not originated by routers into their connected stub areas.

Link-State Advertisement Types for OSPFv3

Each of the following LSA types has a different purpose:

- Router LSA (Type 1)—Describes the link state and costs of a the router link to the area. These LSAs are flooded within an area only. The LSA indicates whether the router is an ABR or ASBR and if it is one end of a virtual link. Type 1 LSAs are also used to advertise stub networks. In OSPFv3, these LSAs have no address information and are network protocol independent. In OSPFv3, router interface information may be spread across multiple router LSAs. Receivers must concatenate all router LSAs originated by a given router before running the SPF calculation.
- Network LSA (Type 2)—Describes the link state and cost information for all routers attached to a multiaccess network segment. This LSA lists all OSPF routers that have interfaces attached to the network segment. Only the elected designated router for the network segment can generate and track the network LSA for the segment. In OSPFv3, network LSAs have no address information and are network-protocol-independent.
- Interarea-prefix LSA for ABRs (Type 3)—Advertises internal networks to routers in other areas (interarea routes). Type 3 LSAs may represent a single network or set of networks aggregated into one prefix. Only ABRs generate Type 3 LSAs. In OSPFv3, addresses for these LSAs are expressed as “prefix and prefix length” instead of “address and mask.” The default route is expressed as a prefix with length 0.
- Interarea-router LSA for ASBRs (Type 4)—Advertises an ASBR and the cost to reach it. Routers that are trying to reach an external network use these advertisements to determine the best path to the next hop. ABRs generate Type 4 LSAs.
- Autonomous system external LSA (Type 5)—Redistributes routes from another autonomous system, usually from a different routing protocol into OSPF. In OSPFv3, addresses for these LSAs are expressed as “prefix and prefix length” instead of “address and mask.” The default route is expressed as a prefix with length 0.
- Autonomous system external LSA (Type 7)—Provides for carrying external route information within an NSSA. Type 7 LSAs may be originated by and advertised throughout an NSSA. NSSAs do not receive

or originate Type 5 LSAs. Type 7 LSAs are advertised only within a single NSSA. They are not flooded into the backbone area or into any other area by border routers.

- Link LSA (Type 8)—Has link-local flooding scope and is never flooded beyond the link with which it is associated. Link LSAs provide the link-local address of the router to all other routers attached to the link or network segment, inform other routers attached to the link of a list of IPv6 prefixes to associate with the link, and allow the router to assert a collection of Options bits to associate with the network LSA that is originated for the link.
- Intra-area-prefix LSAs (Type 9)—A router can originate multiple intra-area-prefix LSAs for every router or transit network, each with a unique link-state ID. The link-state ID for each intra-area-prefix LSA describes its association to either the router LSA or network LSA and contains prefixes for stub and transit networks.

An address prefix occurs in almost all newly defined LSAs. The prefix is represented by three fields: Prefix Length, Prefix Options, and Address Prefix. In OSPFv3, addresses for these LSAs are expressed as “prefix and prefix length” instead of “address and mask.” The default route is expressed as a prefix with length 0.

Inter-area-prefix and intra-area-prefix LSAs carry all IPv6 prefix information that, in IPv4, is included in router LSAs and network LSAs. The Options field in certain LSAs (router LSAs, network LSAs, interarea-router LSAs, and link LSAs) has been expanded to 24 bits to provide support for OSPF in IPv6.

In OSPFv3, the sole function of link-state ID in interarea-prefix LSAs, interarea-router LSAs, and autonomous system external LSAs is to identify individual pieces of the link-state database. All addresses or router IDs that are expressed by the link-state ID in OSPF Version 2 are carried in the body of the LSA in OSPFv3.

Virtual Link and Transit Area for OSPF

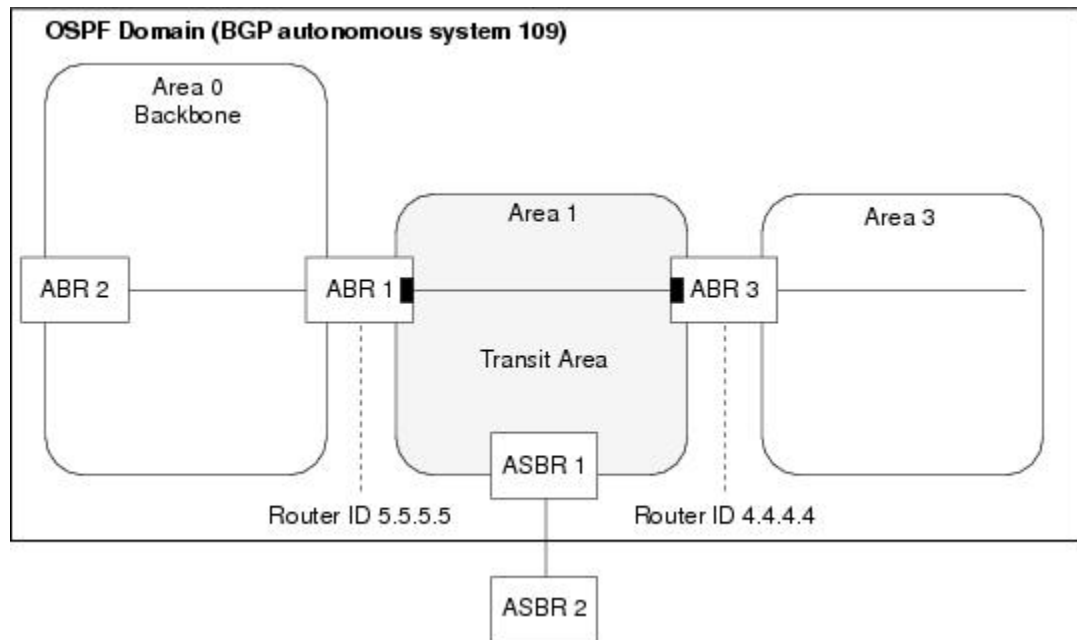
In OSPF, routing information from all areas is first summarized to the backbone area by ABRs. The same ABRs, in turn, propagate such received information to their attached areas. Such hierarchical distribution of routing information requires that all areas be connected to the backbone area (Area 0). Occasions might exist for which an area must be defined, but it cannot be physically connected to Area 0. Examples of such an occasion might be if your company makes a new acquisition that includes an OSPF area, or if Area 0 itself is partitioned.

In the case in which an area cannot be connected to Area 0, you must configure a virtual link between that area and Area 0. The two endpoints of a virtual link are ABRs, and the virtual link must be configured in both routers. The common nonbackbone area to which the two routers belong is called a transit area. A virtual link specifies the transit area and the router ID of the other virtual endpoint (the other ABR).

A virtual link cannot be configured through a stub area or NSSA.

Figure 21: Virtual Link to Area 0

This figure illustrates a virtual link from Area 3 to Area 0.



Passive Interface

Setting an interface as passive disables the sending of routing updates for the neighbors, hence adjacencies will not be formed in OSPF. However, the particular subnet will continue to be advertised to OSPF neighbors. Use the **passive** command in appropriate mode to suppress the sending of OSPF protocol operation on an interface.

It is recommended to use passive configuration on interfaces that are connecting LAN segments with hosts to the rest of the network, but are not meant to be transit links between routers.

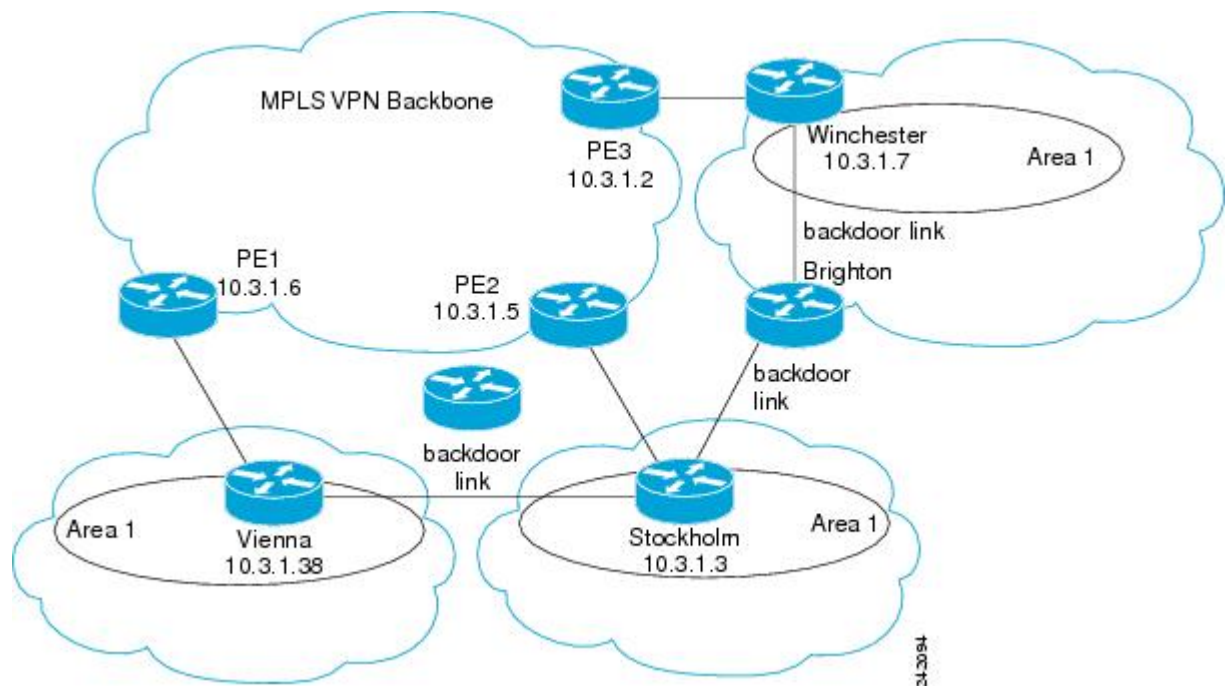
OSPFv2 Sham Link Support for MPLS VPN

In an MPLS VPN environment, several VPN client sites can be connected in the same OSPF area. If these sites are connected over a backdoor link (intra-area link) and connected over the VPN backbone, all traffic passes over the backdoor link instead of over the VPN backbone, because provider edge routers advertise OSPF routes learned over the VPN backbone as inter-area or external routes that are less preferred than intra-area routes advertised over backdoor links.

To correct this default OSPF behavior in an MPLS VPN, configure a sham link between two provider edge (PE) routers to connect the sites through the MPLS VPN backbone. A sham link represents an intra-area (unnumbered point-to-point) connection between PE routers. All other routers in the area see the sham link and use it to calculate intra-area shortest path first (SPF) routes to the remote site. A cost must be configured with each sham link to determine whether traffic is sent over the backdoor link or sham link.

Configured source and destination addresses serve as the endpoints of the sham link. The source and destination IP addresses must belong to the VRF and must be advertised by Border Gateway Protocol (BGP) as host routes to remote PE routers. The sham-link endpoint addresses should not be advertised by OSPF.

Figure 22: Backdoor Paths Between OSPF Client Sites



For example, [Figure 22: Backdoor Paths Between OSPF Client Sites](#), on page 370 shows three client sites, each with backdoor links. Because each site runs OSPF within Area 1 configuration, all routing between the sites follows the intra-area path across the backdoor links instead of over the MPLS VPN backbone.

If the backdoor links between the sites are used only for backup purposes, default route selection over the backbone link is not acceptable as it creates undesirable traffic flow. To establish the desired path selection over the MPLS backbone, an additional OSPF intra-area (sham link) link between the ingress and egress PErouters must be created.

A sham link is required between any two VPN sites that belong to the same OSPF area and share an OSPF backdoor link. If no backdoor link exists between sites, no sham link is required.

Figure 23: Sham Link Between PE Routers to Connected OSPF Client Sites

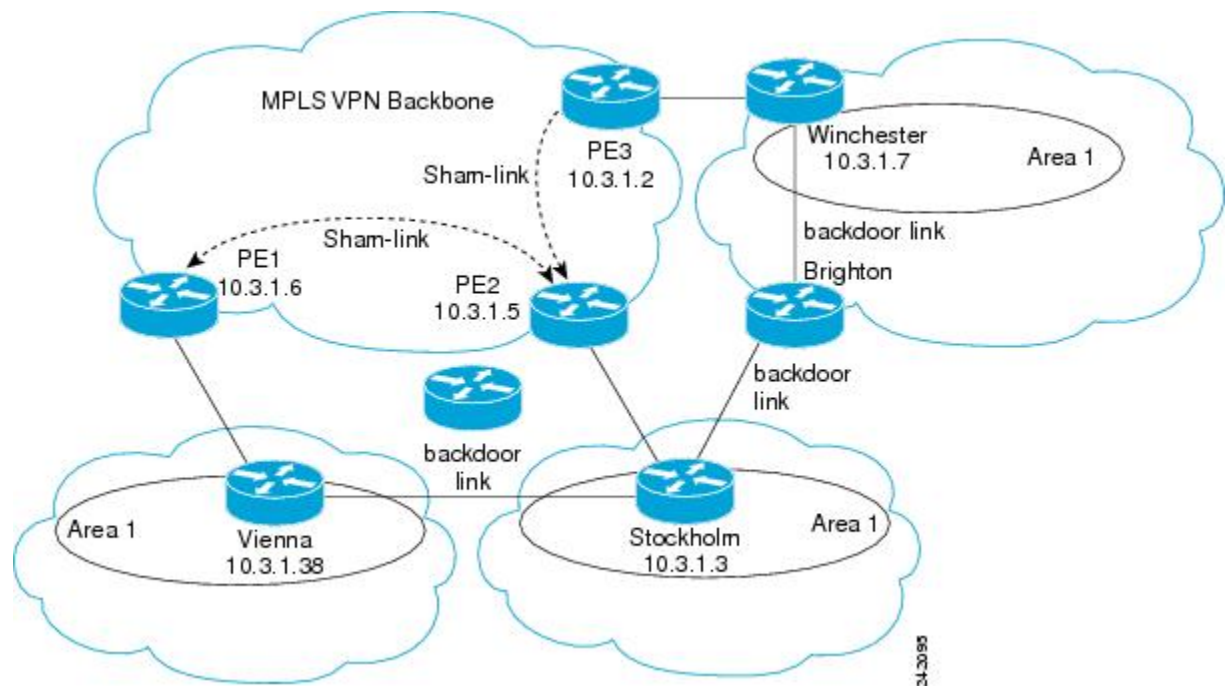


Figure 23: Sham Link Between PE Routers to Connected OSPF Client Sites , on page 371 shows an MPLS VPN topology where a sham link configuration is necessary. A VPN client has three sites, each with a backdoor link. Two sham links are configured, one between PE-1 and PE-2 and another between PE-2 and PE-3. A sham link is not required between PE-1 and PE-3, because there is no backdoor link between these sites.

When a sham link is configured between the PE routers, the PE routers can populate the virtual routing and forwarding (VRF) table with the OSPF routes learned over the sham link. These OSPF routes have a larger administrative distance than BGP routes. If BGP routes are available, they are preferred over these OSPF routes with the high administrative distance.

OSPF SPF Prefix Prioritization

The OSPF SPF Prefix Prioritization feature enables an administrator to converge, in a faster mode, important prefixes during route installation.

When a large number of prefixes must be installed in the Routing Information Base (RIB) and the Forwarding Information Base (FIB), the update duration between the first and last prefix, during SPF, can be significant.

In networks where time-sensitive traffic (for example, VoIP) may transit to the same router along with other traffic flows, it is important to prioritize RIB and FIB updates during SPF for these time-sensitive prefixes.

The OSPF SPF Prefix Prioritization feature provides the administrator with the ability to prioritize important prefixes to be installed, into the RIB during SPF calculations. Important prefixes converge faster among prefixes of the same route type per area. Before RIB and FIB installation, routes and prefixes are assigned to various priority batch queues in the OSPF local RIB, based on specified route policy. The RIB priority batch queues are classified as "critical," "high," "medium," and "low," in the order of decreasing priority.

When enabled, prefix alters the sequence of updating the RIB with this prefix priority:

Critical > High > Medium > Low

As soon as prefix priority is configured, /32 prefixes are no longer preferred by default; they are placed in the low-priority queue, if they are not matched with higher-priority policies. Route policies must be devised to retain /32s in the higher-priority queues (high-priority or medium-priority queues).

Priority is specified using route policy, which can be matched based on IP addresses or route tags. During SPF, a prefix is checked against the specified route policy and is assigned to the appropriate RIB batch priority queue.

These are examples of this scenario:

- If only high-priority route policy is specified, and no route policy is configured for a medium priority:
 - Permitted prefixes are assigned to a high-priority queue.
 - Unmatched prefixes, including /32s, are placed in a low-priority queue.
- If both high-priority and medium-priority route policies are specified, and no maps are specified for critical priority:
 - Permitted prefixes matching high-priority route policy are assigned to a high-priority queue.
 - Permitted prefixes matching medium-priority route policy are placed in a medium-priority queue.
 - Unmatched prefixes, including /32s, are moved to a low-priority queue.
- If both critical-priority and high-priority route policies are specified, and no maps are specified for medium priority:
 - Permitted prefixes matching critical-priority route policy are assigned to a critical-priority queue.
 - Permitted prefixes matching high-priority route policy are assigned to a high-priority queue.
 - Unmatched prefixes, including /32s, are placed in a low-priority queue.
- If only medium-priority route policy is specified and no maps are specified for high priority or critical priority:
 - Permitted prefixes matching medium-priority route policy are assigned to a medium-priority queue.
 - Unmatched prefixes, including /32s, are placed in a low-priority queue.

Use the **[no] spf prefix-priority route-policy *rpl*** command to prioritize OSPF prefix installation into the global RIB during SPF.

SPF prefix prioritization is disabled by default. In disabled mode, /32 prefixes are installed into the global RIB, before other prefixes. If SPF prioritization is enabled, routes are matched against the route-policy criteria and are assigned to the appropriate priority queue based on the SPF priority set. Unmatched prefixes, including /32s, are placed in the low-priority queue.

If all /32s are desired in the high-priority queue or medium-priority queue, configure this single route map:

```
prefix-set ospf-medium-prefixes
 0.0.0.0/0 ge 32
end-set
```

Route Redistribution for OSPF

Redistribution allows different routing protocols to exchange routing information. This technique can be used to allow connectivity to span multiple routing protocols. It is important to remember that the **redistribute** command controls redistribution *into* an OSPF process and not from OSPF. See [Configuration Examples for Implementing OSPF](#), on page 435 for an example of route redistribution for OSPF.

OSPF Shortest Path First Throttling

OSPF SPF throttling makes it possible to configure SPF scheduling in millisecond intervals and to potentially delay SPF calculations during network instability. SPF is scheduled to calculate the Shortest Path Tree (SPT) when there is a change in topology. One SPF run may include multiple topology change events.

The interval at which the SPF calculations occur is chosen dynamically and based on the frequency of topology changes in the network. The chosen interval is within the boundary of the user-specified value ranges. If network topology is unstable, SPF throttling calculates SPF scheduling intervals to be longer until topology becomes stable.

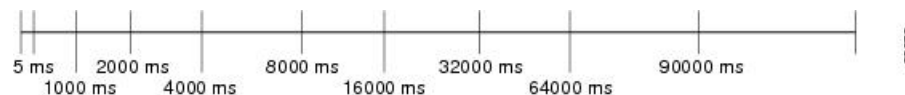
SPF calculations occur at the interval set by the **timers throttle spf** command. The wait interval indicates the amount of time to wait until the next SPF calculation occurs. Each wait interval after that calculation is twice as long as the previous interval until the interval reaches the maximum wait time specified.

The SPF timing can be better explained using an example. In this example, the start interval is set at 5 milliseconds (ms), initial wait interval at 1000 ms, and maximum wait time at 90,000 ms.

```
timers spf 5 1000 90000
```

Figure 24: SPF Calculation Intervals Set by the timers spf Command

This figure shows the intervals at which the SPF calculations occur as long as at least one topology change event is received in a given wait interval.

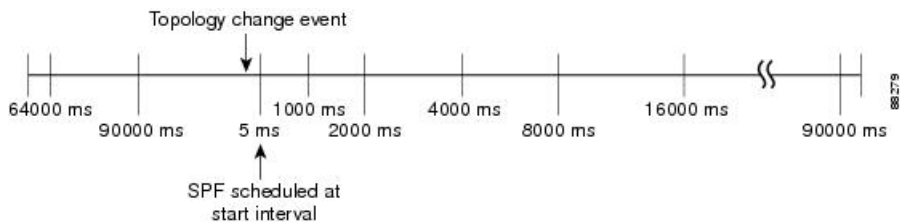


Notice that the wait interval between SPF calculations doubles when at least one topology change event is received during the previous wait interval. After the maximum wait time is reached, the wait interval remains the same until the topology stabilizes and no event is received in that interval.

If the first topology change event is received after the current wait interval, the SPF calculation is delayed by the amount of time specified as the start interval. The subsequent wait intervals continue to follow the dynamic pattern.

If the first topology change event occurs after the maximum wait interval begins, the SPF calculation is again scheduled at the start interval and subsequent wait intervals are reset according to the parameters specified in the **timers throttle spf** command. Notice in [Figure 25: Timer Intervals Reset After Topology Change Event](#), on page 374 that a topology change event was received after the start of the maximum wait time interval and that the SPF intervals have been reset.

Figure 25: Timer Intervals Reset After Topology Change Event



Nonstop Forwarding for OSPF Version 2

Cisco IOS XR Software NSF for OSPF Version 2 allows for the forwarding of data packets to continue along known routes while the routing protocol information is being restored following a process restart or failover. With NSF, peer networking devices do not experience routing flaps. During process restart or failover, data traffic is forwarded through intelligent line cards while the standby Route Processor (RP) assumes control from the failed RP. The ability of line cards to remain up through a process restart, and to be kept current with the Forwarding Information Base (FIB) on the active RP is key to Cisco IOS XR Software NSF operation.

Routing protocols, such as OSPF, run only on the active RP or DRP and receive routing updates from their neighbor routers. When an OSPF NSF-capable router performs a process restart, it must perform two tasks to resynchronize its link-state database with its OSPF neighbors. First, it must relearn the available OSPF neighbors on the network without causing a reset of the neighbor relationship. Second, it must reacquire the contents of the link-state database for the network.

As quickly as possible after an RP failover or process restart, the NSF-capable router sends an OSPF NSF signal to neighboring NSF-aware devices. This signal is in the form of a link-local LSA generated by the failed-over router. Neighbor networking devices recognize this signal as a cue that the neighbor relationship with this router should not be reset. As the NSF-capable router receives signals from other routers on the network, it can begin to rebuild its neighbor list.

After neighbor relationships are reestablished, the NSF-capable router begins to resynchronize its database with all of its NSF-aware neighbors. At this point, the routing information is exchanged between the OSPF neighbors. After this exchange is completed, the NSF-capable device uses the routing information to remove stale routes, update the RIB, and update the FIB with the new forwarding information. OSPF on the router and the OSPF neighbors are now fully converged.

Graceful Shutdown for OSPFv3

The OSPFv3 Graceful Shutdown feature preserves the data plane capability in these circumstances:

- RP failure resulting in a switch-over to the backup processor
- Planned OSPFv3 process restart, such as a restart resulting from a software upgrade or downgrade
- Unplanned OSPFv3 process restart, such as a restart resulting from a process crash

In addition, OSPFv3 will unilaterally shutdown and enter the exited state when a critical memory event, indicating the processor is critically low on available memory, is received from the sysmon watch dog process.

This feature supports nonstop data forwarding on established routes while the OSPFv3 routing protocol restarts. Therefore, this feature enhances high availability of IPv6 forwarding.

Modes of Graceful Restart Operation

The operational modes that a router can be in for this feature are restart mode, helper mode, and protocol shutdown mode.

Restart Mode

When the OSPFv3 process starts up, it determines whether it must attempt a graceful restart. The determination is based on whether graceful restart was previously enabled. (OSPFv3 does not attempt a graceful restart upon the first-time startup of the router.) When OSPFv3 graceful restart is enabled, it changes the purge timer in the RIB to a nonzero value. See [Configuring OSPFv3 Graceful Restart, on page 408](#), for descriptions of how to enable and configure graceful restart.

During a graceful restart, the router does not populate OSPFv3 routes in the RIB. It tries to bring up full adjacencies with the fully adjacent neighbors that OSPFv3 had before the restart. Eventually, the OSPFv3 process indicates to the RIB that it has converged, either for the purpose of terminating the graceful restart (for any reason) or because it has completed the graceful restart.

The following are general details about restart mode. More detailed information on behavior and certain restrictions and requirements appears in [Graceful Restart Requirements and Restrictions, on page 376](#) section.

- If OSPFv3 attempts a restart too soon after the most recent restart, the OSPFv3 process is most likely crashing repeatedly, so the new graceful restart stops running. To control the period between allowable graceful restarts, use the **graceful-restart interval** command.
- When OSPFv3 starts a graceful restart with the first interface that comes up, a timer starts running to limit the duration (or lifetime) of the graceful restart. You can configure this period with the **graceful-restart lifetime** command. On each interface that comes up, a *grace* LSA (Type 11) is flooded to indicate to the neighboring routers that this router is attempting graceful restart. The neighbors enter into helper mode.
- The designated router and backup designated router check of the hello packet received from the restarting neighbor is bypassed, because it might not be valid.

Helper Mode

Helper mode is enabled by default. When a (helper) router receives a grace LSA (Type 11) from a router that is attempting a graceful restart, the following events occur:

- If helper mode has been disabled through the **graceful-restart helper disable** command, the router drops the LSA packet.
- If helper mode is enabled, the router enters helper mode if all of the following conditions are met:
 - The local router itself is not attempting a graceful restart.
 - The local (helping) router has full adjacency with the sending neighbor.
 - The value of *lsage* (link state age) in the received LSA is less than the requested grace period.
 - The sender of the grace LSA is the same as the originator of the grace LSA.
- Upon entering helper mode, a router performs its helper function for a specific period of time. This time period is the lifetime value from the router that is in restart mode—minus the value of *lsage* in the received grace LSA. If the graceful restart succeeds in time, the helper's timer is stopped before it expires. If the helper's timer does expire, the adjacency to the restarting router is brought down, and normal OSPFv3 functionality resumes.

- The dead timer is not honored by the router that is in helper mode.
- A router in helper mode ceases to perform the helper function in any of the following cases:
 - The helper router is able to bring up a FULL adjacency with the restarting router.
 - The local timer for the helper function expires.

Graceful Restart Requirements and Restrictions

The requirements for supporting the Graceful Restart feature include:

- Cooperation of a router's neighbors during a graceful restart. In relation to the router on which OSPFv3 is restarting, each router is called a *helper*.
- All neighbors of the router that does a graceful restart must be capable of doing a graceful restart.
- A graceful restart does not occur upon the first-time startup of a router.
- OSPFv3 neighbor information and database information are not check-pointed.
- An OSPFv3 process rebuilds adjacencies after it restarts.
- To ensure consistent databases after a restart, the OSPFv3 configuration must be identical to the configuration before the restart. (This requirement applies to self-originated information in the local database.) A graceful restart can fail if configurations change during the operation. In this case, data forwarding would be affected. OSPFv3 resumes operation by regenerating all its LSAs and resynchronizing its database with all its neighbors.
- Although IPv6 FIB tables remain unchanged during a graceful restart, these tables eventually mark the routes as stale through the use of a holddown timer. Enough time is allowed for the protocols to rebuild state information and converge.
- The router on which OSPFv3 is restarting must send OSPFv3 hellos within the dead interval of the process restart. Protocols must be able to retain adjacencies with neighbors before the adjacency dead timer expires. The default for the dead timer is 40 seconds. If hellos do not arrive on the adjacency before the dead timer expires, the router takes down the adjacency. The OSPFv3 Graceful Restart feature does not function properly if the dead timer is configured to be less than the time required to send hellos after the OSPFv3 process restarts.
- Simultaneous graceful restart sessions on multiple routers are not supported on a single network segment. If a router determines that multiple routers are in restart mode, it terminates any local graceful restart operation.
- This feature utilizes the available support for changing the purge time of existing OSPFv3 routes in the Routing Information Base (RIB). When graceful restart is enabled, the purge timer is set to 90 seconds by default. If graceful restart is disabled, the purge timer setting is 0.
- This feature has an associated *grace* LSA. This link-scope LSA is type11.
- According to the RFC, the OSPFv3 process should flush all old, self-originated LSAs during a restart. With the Graceful Restart feature, however, the router delays this flushing of unknown self-originated LSAs during a graceful restart. OSPFv3 can learn new information and build new LSAs to replace the old LSAs. When the delay is over, all old LSAs are flushed.

- If graceful restart is enabled, the adjacency creation time of all the neighbors is saved in the system database (SysDB). The purpose for saving the creation time is so that OSPFv3 can use the original adjacency creation time to display the uptime for that neighbor after the restart.

Warm Standby and Nonstop Routing for OSPF Version 2

OSPFv2 warm standby provides high availability across RP switchovers. With warm standby extensions, each process running on the active RP has a corresponding standby process started on the standby RP. A standby OSPF process can send and receive OSPF packets with no performance impact to the active OSPF process.

Nonstop routing (NSR) allows an RP failover, process restart, or in-service upgrade to be invisible to peer routers and ensures that there is minimal performance or processing impact. Routing protocol interactions between routers are not impacted by NSR. NSR is built on the warm standby extensions. NSR alleviates the requirement for Cisco NSF and IETF graceful restart protocol extensions.



Note It is recommended to set the hello timer interval to the default of 10 seconds. OSPF sessions may flap during switchover if hello-interval timer configured is less than default value.

Warm Standby for OSPF Version 3

This feature helps OSPFv3 to initialize itself prior to Fail over (FO) and be ready to function before the failure occurs. It reduces the downtime during switchover. By default, the router sends hello packets every 40 seconds.

With warm standby process for each OSPF process running on the Active Route Processor, the corresponding OSPF process must start on the Standby RP. There are no changes in configuration for this feature.

Warm-Standby is always enabled. This is an advantage for the systems running OSPFv3 as their IGP when they do RP failover.

Multicast-Intact Support for OSPF

The multicast-intact feature provides the ability to run multicast routing (PIM) when IGP shortcuts are configured and active on the router. Both OSPFv2 and IS-IS support the multicast-intact feature.

You can enable multicast-intact in the IGP when multicast routing protocols (PIM) are configured and IGP shortcuts are configured on the router. IGP shortcuts are MPLS tunnels that are exposed to IGP. The IGP routes IP traffic over these tunnels to destinations that are downstream from the egress router of the tunnel (from an SPF perspective). PIM cannot use IGP shortcuts for propagating PIM joins, because reverse path forwarding (RPF) cannot work across a unidirectional tunnel.

When you enable multicast-intact on an IGP, the IGP publishes a parallel or alternate set of equal-cost next hops for use by PIM. These next hops are called *mcast-intact* next hops. The *mcast-intact* next hops have the following attributes:

- They are guaranteed not to contain any IGP shortcuts.
- They are not used for unicast routing but are used only by PIM to look up an IPv4 next-hop to a PIM source.

- They are not published to the FIB.
- When multicast-intact is enabled on an IGP, all IPv4 destinations that were learned through link-state advertisements are published with a set equal-cost multicast-intact next hops to the RIB. This attribute applies even when the native next hops have no IGP shortcuts.

In OSPF, the max-paths (number of equal-cost next hops) limit is applied separately to the native and multicast-intact next hops. The number of equal cost multicast-intact next hops is the same as that configured for the native next hops.

Load Balancing in OSPF Version 2 and OSPFv3

When a router learns multiple routes to a specific network by using multiple routing processes (or routing protocols), it installs the route with the lowest administrative distance in the routing table. Sometimes the router must select a route from among many learned by using the same routing process with the same administrative distance. In this case, the router chooses the path with the lowest cost (or metric) to the destination. Each routing process calculates its cost differently; the costs may need to be manipulated to achieve load balancing.

OSPF performs load balancing automatically. If OSPF finds that it can reach a destination through more than one interface and each path has the same cost, it installs each path in the routing table. The only restriction on the number of paths to the same destination is controlled by the **maximum-paths** (OSPF) command.

The range for maximum paths is 1 to 32 and the default number of maximum paths is 32.

Multi-Area Adjacency for OSPF Version 2

The multi-area adjacency feature for OSPFv2 allows a link to be configured on the primary interface in more than one area so that the link could be considered as an intra-area link in those areas and configured as a preference over more expensive paths.

This feature establishes a point-to-point unnumbered link in an OSPF area. A point-to-point link provides a topological path for that area, and the primary adjacency uses the link to advertise the link consistent with draft-ietf-ospf-multi-area-adj-06.

The following are multi-area interface attributes and limitations:

- Exists as a logical construct over an existing primary interface for OSPF; however, the neighbor state on the primary interface is independent of the multi-area interface.
- Establishes a neighbor relationship with the corresponding multi-area interface on the neighboring router. A mixture of multi-area and primary interfaces is not supported.
- Advertises an unnumbered point-to-point link in the router link state advertisement (LSA) for the corresponding area when the neighbor state is full.
- Created as a point-to-point network type. You can configure multi-area adjacency on any interface where only two OSPF speakers are attached. In the case of native broadcast networks, the interface must be configured as an OSPF point-to-point type using the **network point-to-point** command to enable the interface for a multi-area adjacency.
- Inherits the Bidirectional Forwarding Detection (BFD) characteristics from its primary interface. BFD is not configurable under a multi-area interface; however, it is configurable under the primary interface.

The multi-area interface inherits the interface characteristics from its primary interface, but some interface characteristics can be configured under the multi-area interface configuration mode as shown below:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# multi-area-interface GigabitEthernet 0/1/0/3
RP/0/RP0/CPU0:router(config-ospf-ar-mif)# ?
 authentication          Enable authentication
 authentication-key      Authentication password (key)
 cost                   Interface cost
 cost-fallback           Cost when cumulative bandwidth goes below the threshold
 database-filter        Filter OSPF LSA during synchronization and flooding
 dead-interval          Interval after which a neighbor is declared dead
 distribute-list         Filter networks in routing updates
 hello-interval         Time between HELLO packets
 message-digest-key     Message digest authentication password (key)
 mtu-ignore             Enable/Disable ignoring of MTU in DBD packets
 packet-size            Customize size of OSPF packets upto MTU
 retransmit-interval    Time between retransmitting lost link state advertisements
 transmit-delay         Estimated time needed to send link-state update packet

RP/0/RP0/CPU0:router(config-ospf-ar-mif)#
```

Label Distribution Protocol IGP Auto-configuration for OSPF

Label Distribution Protocol (LDP) Interior Gateway Protocol (IGP) auto-configuration simplifies the procedure to enable LDP on a set of interfaces used by an IGP instance, such as OSPF. LDP IGP auto-configuration can be used on a large number of interfaces (for example, when LDP is used for transport in the core) and on multiple OSPF instances simultaneously.

This feature supports the IPv4 unicast address family for the default VPN routing and forwarding (VRF) instance.

LDP IGP auto-configuration can also be explicitly disabled on an individual interface basis under LDP using the **ldp auto-config disable** command. This allows LDP to receive all OSPF interfaces minus the ones explicitly disabled.

See *MPLS Configuration Guide for the Cisco CRS Routers* for information on configuring LDP IGP auto-configuration.

OSPF Authentication Message Digest Management

All OSPF routing protocol exchanges are authenticated and the method used can vary depending on how authentication is configured. When using cryptographic authentication, the OSPF routing protocol uses the Message Digest 5 (MD5) authentication algorithm to authenticate packets transmitted between neighbors in the network. For each OSPF protocol packet, a key is used to generate and verify a message digest that is appended to the end of the OSPF packet. The message digest is a one-way function of the OSPF protocol packet and the secret key. Each key is identified by the combination of interface used and the key identification. An interface may have multiple keys active at any time.

To manage the rollover of keys and enhance MD5 authentication for OSPF, you can configure a container of keys called a *keychain* with each key comprising the following attributes: generate/accept time, key identification, and authentication algorithm.

GTSM TTL Security Mechanism for OSPF

OSPF is a link state protocol that requires networking devices to detect topological changes in the network, flood Link State Advertisement (LSA) updates to neighbors, and quickly converge on a new view of the topology. However, during the act of receiving LSAs from neighbors, network attacks can occur, because there are no checks that unicast or multicast packets are originating from a neighbor that is one hop away or multiple hops away over virtual links.

For virtual links, OSPF packets travel multiple hops across the network; hence, the TTL value can be decremented several times. For these type of links, a minimum TTL value must be allowed and accepted for multiple-hop packets.

To filter network attacks originating from invalid sources traveling over multiple hops, the Generalized TTL Security Mechanism (GTSM), RFC 3682, is used to prevent the attacks. GTSM filters link-local addresses and allows for only one-hop neighbor adjacencies through the configuration of TTL value 255. The TTL value in the IP header is set to 255 when OSPF packets are originated, and checked on the received OSPF packets against the default GTSM TTL value 255 or the user configured GTSM TTL value, blocking unauthorized OSPF packets originated from TTL hops away.

Path Computation Element for OSPFv2

A PCE is an entity (component, application, or network node) that is capable of computing a network path or route based on a network graph and applying computational constraints.

PCE is accomplished when a PCE address and client is configured for MPLS-TE. PCE communicates its PCE address and capabilities to OSPF then OSPF packages this information in the PCE Discovery type-length-value (TLV) (Type 2) and reoriginates the RI LSA. OSPF also includes the Router Capabilities TLV (Type 1) in all its RI LSAs. The PCE Discovery TLV contains the PCE address sub-TLV (Type 1) and the Path Scope Sub-TLV (Type 2).

The PCE Address Sub-TLV specifies the IP address that must be used to reach the PCE. It should be a loop-back address that is always reachable, this TLV is mandatory, and must be present within the PCE Discovery TLV. The Path Scope Sub-TLV indicates the PCE path computation scopes, which refers to the PCE ability to compute or participate in the computation of intra-area, inter-area, inter-AS or inter-layer TE LSPs.

PCE extensions to OSPFv2 include support for the Router Information Link State Advertisement (RI LSA). OSPFv2 is extended to receive all area scopes (LSA Types 9, 10, and 11). However, OSPFv2 originates only area scope Type 10.

For detailed information for the Path Computation Element feature see the *Implementing MPLS Traffic Engineering on Cisco IOS XR Software* module of the *MPLS Configuration Guide for the Cisco CRS Routers* and the following IETF drafts:

- draft-ietf-ospf-cap-09
- draft-ietf-pce-disco-proto-ospf-00

OSPF Queue Tuning Parameters

The OSPF queue tuning parameters configuration allows you to:

- Limit the number of continuous incoming events processed.

- Set the maximum number of rate-limited link-state advertisements (LSAs) processed per run.
- Limit the number of summary or external Type 3 to Type 7 link-state advertisements (LSAs) processed per shortest path first (SPF) iteration within a single SPF run.
- Set the high watermark for incoming priority events.

OSPF IP Fast Reroute Loop Free Alternate

The OSPF IP Fast Reroute (FRR) Loop Free Alternate (LFA) computation supports these:

- Fast rerouting capability by using IP forwarding and routing
- Handles failure in the line cards in minimum time

OSPF Over GRE Interfaces

Cisco IOS XR software provides the capability to run OSPF protocols over Generic Routing Encapsulation (GRE) tunnel interfaces.

For more information on GRE tunnel interfaces, see *Implementing BGP on Cisco IOS XR Software* module.

Management Information Base (MIB) for OSPFv3

Cisco IOS XR supports full MIBs and traps for OSPFv3, as defined in RFC 5643. The RFC 5643 defines objects of the Management Information Base (MIB) for use with the Open Shortest Path First (OSPF) Routing Protocol for IPv6 (OSPF version 3).

The OSPFv3 MIB implementation is based on the IETF draft *Management Information Base for OSPFv3 (draft-ietf-ospf-ospfv3-mib-8)*. Users need to update the NMS application to pick up the new MIB when upgraded to RFC 5643.

Refer to the *Cisco Carrier Routing System and Cisco XR 12000 Series Router MIB Support Guide* for more information on Cisco IOS XR MIB support.

Multiple OSPFv3 Instances

SNMPv3 supports "contexts" that can be used to implement MIB views on multiple OSPFv3 instances, in the same system.

VRF-lite Support for OSPFv2

VRF-lite capability is enabled for OSPF version 2 (OSPFv2). VRF-lite is the virtual routing and forwarding (VRF) deployment without the BGP/MPLS based backbone. In VRF-lite, individual provider edge (PE) routers are directly connected using VRF interfaces. To enable VRF-lite in OSPFv2, configure the **capability vrf-lite** command in VRF configuration mode. When VRF-lite is configured, the DN bit processing and the automatic Area Border Router (ABR) status setting are disabled.

OSPFv3 Timers Link-state Advertisements and Shortest Path First Throttle Default Values Update

The Open Shortest Path First version 3 (OSPFv3) timers link-state advertisements (LSAs) and shortest path first (SPF) throttle default values are updated to:

- **timers throttle lsa all**—*start-interval*: 50 milliseconds and *hold-interval*: 200 milliseconds
- **timers throttle spf**—*spf-start*: 50 milliseconds, *spf-hold*: 200 milliseconds, *spf-max-wait*: 5000 milliseconds

Unequal Cost Multipath Load-balancing for OSPF

The unequal cost multipath (UCMP) load-balancing adds the capability with Open Shortest Path First (OSPF) to load-balance traffic proportionally across multiple paths, with different cost. Without UCMP enabled, only the best cost paths are discovered by OSPF (ECMP) and alternate higher cost paths are not computed.

Generally, higher bandwidth links have lower IGP metrics configured, so that they form the shortest IGP paths. With the UCMP load-balancing enabled, IGP can use even lower bandwidth links or higher cost links for traffic, and can install these paths to the forwarding information base (FIB). OSPF installs multiple paths to the same destination in FIB, but each path will have a 'load metric/weight' associated with it. FIB uses this load metric/weight to decide the amount of traffic that needs to be sent on a higher bandwidth path and the amount of traffic that needs to be sent on a lower bandwidth path.

The UCMP computation is provided under OSPF VRF context, enabling UCMP computation for a particular VRF. For default VRF the configuration is done under the OSPF global mode. The UCMP configuration is also provided with a prefix-list option, which would limit the UCMP computation only for the prefixes present in the prefix-list. If prefix-list option is not provided, UCMP computation is done for the reachable prefixes in OSPF. The number of UCMP paths to be considered and installed is controlled using the **variance** configuration. Variance value identifies the range for the UCMP path metric to be considered for installation into routing information base (RIB/FIB) and is defined in terms of a percentage of the primary path metric. Total number of paths, including ECMP and UCMP paths together is limited by the max-path configuration or by the max-path capability of the platform.

There is an option to exclude an interface from being used for UCMP computation. If it is desired that a particular interface should not be considered as a UCMP nexthop, for any prefix, then use the **UCMP exclude interface** command to configure the interface to be excluded from UCMP computation.

Enabling the UCMP configuration indicates that OSPF should perform UCMP computation for the all the reachable OSPF prefixes or all the prefixes permitted by the prefix-list, if the prefix-list option is used. The UCMP computation happens only after the primary SPF and route calculation is completed. There would be a configurable delay (default delay is 100 ms) from the time primary route calculation is completed and UCMP computation is started. Use the **UCMP delay-interval** command to configure the delay between primary SPF completion and start of UCMP computation. UCMP computation will be done during the fast re-route computation (IPFRR does not need to be enabled for UCMP computation to be performed). If IPFRR is enabled, the fast re-route backup paths will be calculated for both the primary equal cost multipath (ECMP) paths and the UCMP paths.

To manually adjust UCMP ratio, use any command that changes the metric of the link.

- By using the bandwidth command in interface configuration mode
- By adjusting the OSPF interface cost on the link

How to Implement OSPF

This section contains the following procedures:

Enabling OSPF

This task explains how to perform the minimum OSPF configuration on your router that is to enable an OSPF process with a router ID, configure a backbone or nonbackbone area, and then assign one or more interfaces on which OSPF runs.

Before you begin

Although you can configure OSPF before you configure an IP address, no OSPF routing occurs until at least one IP address is configured.

SUMMARY STEPS

1. **configure**
2. Do one of the following:
 - **router ospf** *process-name*
 - **router ospfv3** *process-name*
3. **router-id** { *router-id* }
4. **area** *area-id*
5. **interface** *type interface-path-id*
6. Repeat Step 5 for each interface that uses OSPF.
7. **log adjacency changes** [**detail**] [**disable**]
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	Do one of the following: <ul style="list-style-type: none"> • router ospf <i>process-name</i> • router ospfv3 <i>process-name</i> Example: <pre>RP/0/RP0/CPU0:router(config)# router ospf 1</pre> or <pre>RP/0/RP0/CPU0:router(config)# router ospfv3 1</pre>	Enables OSPF routing for the specified routing process and places the router in router configuration mode. or Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	router-id { <i>router-id</i> } Example:	Configures a router ID for the OSPF process.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3	Note We recommend using a stable IP address as the router ID.
Step 4	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config-ospf)# area 0	Enters area configuration mode and configures an area for the OSPF process. <ul style="list-style-type: none"> • Backbone areas have an area ID of 0. • Nonbackbone areas have a nonzero area ID. • The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 5	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/1/0/3	Enters interface configuration mode and associates one or more interfaces for the area configured in Step 4.
Step 6	Repeat Step 5 for each interface that uses OSPF.	—
Step 7	log adjacency changes [detail] [disable] Example: RP/0/RP0/CPU0:router(config-ospf-ar-if)# log adjacency changes detail	(Optional) Requests notification of neighbor changes. <ul style="list-style-type: none"> • By default, this feature is enabled. • The messages generated by neighbor changes are considered notifications, which are categorized as severity Level 5 in the logging console command. The logging console command controls which severity level of messages are sent to the console. By default, all severity level messages are sent.
Step 8	commit	

Configuring Stub and Not-So-Stubby Area Types

This task explains how to configure the stub area and the NSSA for OSPF.

SUMMARY STEPS

1. **configure**
2. Do one of the following:
 - **router ospf** *process-name*
 - **router ospfv3** *process-name*
3. **router-id** { *router-id* }
4. **area** *area-id*

5. Do one of the following:
 - **stub** [**no-summary**]
 - **nssa** [**no-redistribution**] [**default-information-originate**] [**no-summary**] [**translate**] [**translate always**]
6. Do one of the following:
 - **stub**
 - **nssa**
7. **default-cost** *cost*
8. **commit**
9. Repeat this task on all other routers in the stub area or NSSA.

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	Do one of the following: <ul style="list-style-type: none"> • router ospf <i>process-name</i> • router ospfv3 <i>process-name</i> Example: <pre>RP/0/RP0/CPU0:router(config)# router ospf 1</pre> or <pre>RP/0/RP0/CPU0:router(config)# router ospfv3 1</pre>	Enables OSPF routing for the specified routing process and places the router in router configuration mode. or Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	router-id { <i>router-id</i> } Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3</pre>	Configures a router ID for the OSPF process. Note We recommend using a stable IP address as the router ID.
Step 4	area <i>area-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# area 1</pre>	Enters area configuration mode and configures a nonbackbone area for the OSPF process. <ul style="list-style-type: none"> • The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 5	Do one of the following: <ul style="list-style-type: none"> • stub [no-summary] • nssa [no-redistribution] [default-information-originate] [no-summary] [translate] [translate always] 	Defines the nonbackbone area as a stub area. <ul style="list-style-type: none"> • Specify the no-summary keyword to further reduce the number of LSAs sent into a stub area. This keyword prevents the ABR from sending summary link-state advertisements (Type 3) in the stub area.

	Command or Action	Purpose
	<p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# stub no summary</pre> <p>or</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# nssa no-redistribution</pre> <p>or</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# nssa translate <type number> always</pre>	<p>or</p> <p>Defines an area as an NSSA.</p>
Step 6	<p>Do one of the following:</p> <ul style="list-style-type: none"> • stub • nssa <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# stub</pre> <p>or</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# nssa</pre>	<p>(Optional) Turns off the options configured for stub and NSSA areas.</p> <ul style="list-style-type: none"> • If you configured the stub and NSSA areas using the optional keywords (no-summary , no-redistribution , default-information-originate , and translate) in Step 5, you must now reissue the stub and nssa commands without the keywords—rather than using the no form of the command. • For example, the no nssa default-information-originate form of the command changes the NSSA area into a normal area that inadvertently brings down the existing adjacencies in that area.
Step 7	<p>default-cost <i>cost</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)#default-cost 15</pre>	<p>(Optional) Specifies a cost for the default summary route sent into a stub area or an NSSA.</p> <ul style="list-style-type: none"> • Use this command only on ABRs attached to the NSSA. Do not use it on any other routers in the area. • The default cost is 1.
Step 8	commit	
Step 9	Repeat this task on all other routers in the stub area or NSSA.	—

Configuring Neighbors for Nonbroadcast Networks

This task explains how to configure neighbors for a nonbroadcast network. This task is optional.

Before you begin

Configuring NBMA networks as either broadcast or nonbroadcast assumes that there are virtual circuits from every router to every router or fully meshed network.

SUMMARY STEPS

1. **configure**
2. Do one of the following:
 - **router ospf** *process-name*
 - **router ospfv3** *process-name*
3. **router-id** { *router-id* }
4. **area** *area-id*
5. **network** { **broadcast** | **non-broadcast** | { **point-to-multipoint** [**non-broadcast**] | **point-to-point** } }
6. **dead-interval** *seconds*
7. **hello-interval** *seconds*
8. **interface** *type interface-path-id*
9. Do one of the following:
 - **neighbor** *ip-address* [**priority** *number*] [**poll-interval** *seconds*] [**cost** *number*]
 - **neighbor** *ipv6-link-local-address* [**priority** *number*] [**poll-interval** *seconds*] [**cost** *number*] [**database-filter** [**all**]]
10. Repeat Step 9 for all neighbors on the interface.
11. **exit**
12. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	Do one of the following: <ul style="list-style-type: none"> • router ospf <i>process-name</i> • router ospfv3 <i>process-name</i> Example: <pre>RP/0/RP0/CPU0:router(config)# router ospf 1</pre> or <pre>RP/0/RP0/CPU0:router(config)# router ospfv3 1</pre>	Enables OSPF routing for the specified routing process and places the router in router configuration mode. or Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	router-id { <i>router-id</i> } Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3</pre>	Configures a router ID for the OSPF process. Note We recommend using a stable IP address as the router ID.
Step 4	area <i>area-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# area 0</pre>	Enters area configuration mode and configures an area for the OSPF process. <ul style="list-style-type: none"> • The example configures a backbone area.

	Command or Action	Purpose
		<ul style="list-style-type: none"> The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 5	<p>network { broadcast non-broadcast { point-to-multipoint [non-broadcast] point-to-point } }</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# network non-broadcast</pre>	<p>Configures the OSPF network type to a type other than the default for a given medium.</p> <ul style="list-style-type: none"> The example sets the network type to NBMA.
Step 6	<p>dead-interval <i>seconds</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# dead-interval 40</pre>	<p>(Optional) Sets the time to wait for a hello packet from a neighbor before declaring the neighbor down.</p>
Step 7	<p>hello-interval <i>seconds</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# hello-interval 10</pre>	<p>(Optional) Specifies the interval between hello packets that OSPF sends on the interface.</p> <p>Note It is recommended to set the hello timer interval to the default of 10 seconds. OSPF sessions may flap during switchover if hello-interval timer configured is less than default value.</p>
Step 8	<p>interface <i>type interface-path-id</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/2/0/0</pre>	<p>Enters interface configuration mode and associates one or more interfaces for the area configured in Step 4.</p> <ul style="list-style-type: none"> In this example, the interface inherits the nonbroadcast network type and the hello and dead intervals from the areas because the values are not set at the interface level.
Step 9	<p>Do one of the following:</p> <ul style="list-style-type: none"> neighbor <i>ip-address</i> [priority <i>number</i>] [poll-interval <i>seconds</i>] [cost <i>number</i>] neighbor <i>ipv6-link-local-address</i> [priority <i>number</i>] [poll-interval <i>seconds</i>] [cost <i>number</i>] [database-filter [all]] <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar-if)# neighbor 10.20.20.1 priority 3 poll-interval 15</pre> <p>or</p>	<p>Configures the IPv4 address of OSPF neighbors interconnecting to nonbroadcast networks.</p> <p>or</p> <p>Configures the link-local IPv6 address of OSPFv3 neighbors.</p> <ul style="list-style-type: none"> The <i>ipv6-link-local-address</i> argument must be in the form documented in RFC 2373 in which the address is specified in hexadecimal using 16-bit values between colons. The priority keyword notifies the router that this neighbor is eligible to become a DR or BDR. The

	Command or Action	Purpose
	<pre>RP/0/RP0/CPU0:router(config-ospf-ar-if)# neighbor fe80::3203:a0ff:fe9d:f3fe</pre>	<p>priority value should match the actual priority setting on the neighbor router. The neighbor priority default value is zero. This keyword does not apply to point-to-multipoint interfaces.</p> <ul style="list-style-type: none"> The poll-interval keyword does not apply to point-to-multipoint interfaces. RFC 1247 recommends that this value be much larger than the hello interval. The default is 120 seconds (2 minutes). Neighbors with no specific cost configured assumes the cost of the interface, based on the cost command. On point-to-multipoint interfaces, cost number is the only keyword and argument combination that works. The cost keyword does not apply to NBMA networks. The database-filter keyword filters outgoing LSAs to an OSPF neighbor. If you specify the all keyword, incoming and outgoing LSAs are filtered. Use with extreme caution since filtering may cause the routing topology to be seen as entirely different between two neighbors, resulting in 'black-holing' of data traffic or routing loops.
Step 10	Repeat Step 9 for all neighbors on the interface.	—
Step 11	<pre>exit</pre> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar-if)# exit</pre>	Enters area configuration mode.
Step 12	commit	

Configuring Authentication at Different Hierarchical Levels for OSPF Version 2

This task explains how to configure MD5 (secure) authentication on the OSPF router process, configure one area with plain text authentication, and then apply one interface with clear text (null) authentication.



Note Authentication configured at the interface level overrides authentication configured at the area level and the router process level. If an interface does not have authentication specifically configured, the interface inherits the authentication parameter value from a higher hierarchical level. See [OSPF Hierarchical CLI and CLI Inheritance, on page 358](#) for more information about hierarchy and inheritance.

Before you begin

If you choose to configure authentication, you must first decide whether to configure plain text or MD5 authentication, and whether the authentication applies to all interfaces in a process, an entire area, or specific interfaces. See [Route Authentication Methods for OSPF, on page 362](#) for information about each type of authentication and when you should use a specific method for your network.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **router-id** { *router-id* }
4. **authentication** [**message-digest** | **null**]
5. **message-digest-key** *key-id* **md5** { *key* | **clear** *key* | **encrypted** *key* | **LINE** }
6. **area** *area-id*
7. **interface** *type interface-path-id*
8. Repeat Step 7 for each interface that must communicate, using the same authentication.
9. **exit**
10. **area** *area-id*
11. **authentication** [**message-digest** | **null**]
12. **interface** *type interface-path-id*
13. Repeat Step 12 for each interface that must communicate, using the same authentication.
14. **interface** *type interface-path-id*
15. **authentication** [**message-digest** | **null**]
16. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospf 1	Enables OSPF routing for the specified routing process and places the router in router configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	router-id { <i>router-id</i> } Example: RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3	Configures a router ID for the OSPF process.
Step 4	authentication [message-digest null] Example: RP/0/RP0/CPU0:router(config-ospf)#authentication message-digest	Enables MD5 authentication for the OSPF process. • This authentication type applies to the entire router process unless overridden by a lower hierarchical level such as the area or interface.

	Command or Action	Purpose
Step 5	<p>message-digest-key <i>key-id</i> md5 { <i>key</i> clear <i>key</i> encrypted <i>key</i> LINE }</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)#message-digest-key 4 md5 yourkey</pre>	<p>Specifies the MD5 authentication key for the OSPF process.</p> <ul style="list-style-type: none"> The neighbor routers must have the same key identifier.
Step 6	<p>area <i>area-id</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# area 0</pre>	<p>Enters area configuration mode and configures a backbone area for the OSPF process.</p>
Step 7	<p>interface <i>type interface-path-id</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/1/0/3</pre>	<p>Enters interface configuration mode and associates one or more interfaces to the backbone area.</p> <ul style="list-style-type: none"> All interfaces inherit the authentication parameter values specified for the OSPF process (Step 4, Step 5, and Step 6).
Step 8	<p>Repeat Step 7 for each interface that must communicate, using the same authentication.</p>	—
Step 9	<p>exit</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# exit</pre>	<p>Enters area OSPF configuration mode.</p>
Step 10	<p>area <i>area-id</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# area 1</pre>	<p>Enters area configuration mode and configures a nonbackbone area 1 for the OSPF process.</p> <ul style="list-style-type: none"> The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 11	<p>authentication [message-digest null]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# authentication</pre>	<p>Enables Type 1 (plain text) authentication that provides no security.</p> <ul style="list-style-type: none"> The example specifies plain text authentication (by not specifying a keyword). Use the authentication-key command in interface configuration mode to specify the plain text password.
Step 12	<p>interface <i>type interface-path-id</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/1/0/0</pre>	<p>Enters interface configuration mode and associates one or more interfaces to the nonbackbone area 1 specified in Step 7.</p> <ul style="list-style-type: none"> All interfaces configured inherit the authentication parameter values configured for area 1.

	Command or Action	Purpose
Step 13	Repeat Step 12 for each interface that must communicate, using the same authentication.	—
Step 14	interface <i>type interface-path-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/3/0/0</pre>	Enters interface configuration mode and associates one or more interfaces to a different authentication type.
Step 15	authentication [message-digest null] Example: <pre>RP/0/RP0/CPU0:router(config-ospf-ar-if)# authentication null</pre>	Specifies no authentication on GigabitEthernet interface 0/3/0/0, overriding the plain text authentication specified for area 1. <ul style="list-style-type: none"> • By default, all of the interfaces configured in the same area inherit the same authentication parameter values of the area.
Step 16	commit	

Controlling the Frequency That the Same LSA Is Originated or Accepted for OSPF

This task explains how to tune the convergence time of OSPF routes in the routing table when many LSAs need to be flooded in a very short time interval.

SUMMARY STEPS

1. **configure**
2. Do one of the following:
 - **router ospf** *process-name*
 - **router ospfv3** *process-name*
3. **router-id** { *router-id* }
4. Perform Step 5 or Step 6 or both to control the frequency that the same LSA is originated or accepted.
5. **timers lsa refresh** *seconds*
6. **timers lsa min-arrival** *seconds*
7. **timers lsa group-pacing** *seconds*
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	Do one of the following: <ul style="list-style-type: none"> • router ospf <i>process-name</i> • router ospfv3 <i>process-name</i> 	Enables OSPF routing for the specified routing process and places the router in router configuration mode. or

	Command or Action	Purpose
	<p>Example:</p> <pre>RP/0/RP0/CPU0:router:router(config)# router ospf 1 or RP/0/RP0/CPU0:router(config)# router ospfv3 1</pre>	<p>Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.</p> <p>Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.</p>
Step 3	<p>router-id { <i>router-id</i> }</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3</pre>	<p>Configures a router ID for the OSPF process.</p> <p>Note We recommend using a stable IP address as the router ID.</p>
Step 4	Perform Step 5 or Step 6 or both to control the frequency that the same LSA is originated or accepted.	—
Step 5	<p>timers lsa refresh <i>seconds</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# timers lsa refresh 1800</pre>	<p>Sets how often self-originated LSAs should be refreshed, in seconds.</p> <ul style="list-style-type: none"> • The default is 1800 seconds for both OSPF and OSPFv3.
Step 6	<p>timers lsa min-arrival <i>seconds</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# timers lsa min-arrival 2</pre>	<p>Limits the frequency that new processes of any particular OSPF Version 2 LSA can be accepted during flooding.</p> <ul style="list-style-type: none"> • The default is 1 second.
Step 7	<p>timers lsa group-pacing <i>seconds</i></p> <p>Example:</p> <pre>RP/0/ RP0/CPU0:router(config-ospf)# timers lsa group-pacing 1000</pre>	<p>Changes the interval at which OSPF link-state LSAs are collected into a group for flooding.</p> <ul style="list-style-type: none"> • The default is 240 seconds.
Step 8	commit	

Creating a Virtual Link with MD5 Authentication to Area 0 for OSPF

This task explains how to create a virtual link to your backbone (area 0) and apply MD5 authentication. You must perform the steps described on both ABRs, one at each end of the virtual link. To understand virtual links, see [Virtual Link and Transit Area for OSPF, on page 368](#).



Note After you explicitly configure area parameter values, they are inherited by all interfaces bound to that area—unless you override the values and configure them explicitly for the interface. An example is provided in [Virtual Link Configured with MD5 Authentication for OSPF Version 2: Example, on page 440](#).

Before you begin

The following prerequisites must be met before creating a virtual link with MD5 authentication to area 0:

- You must have the router ID of the neighbor router at the opposite end of the link to configure the local router. You can execute the **show ospf** or **show ospfv3** command on the remote router to get its router ID.
- For a virtual link to be successful, you need a stable router ID at each end of the virtual link. You do not want them to be subject to change, which could happen if they are assigned by default. (See [OSPF Process and Router ID, on page 361](#) for an explanation of how the router ID is determined.) Therefore, we recommend that you perform one of the following tasks before configuring a virtual link:
 - Use the **router-id** command to set the router ID. This strategy is preferable.
 - Configure a loopback interface so that the router has a stable router ID.
- Before configuring your virtual link for OSPF Version 2, you must decide whether to configure plain text authentication, MD5 authentication, or no authentication (which is the default). Your decision determines whether you need to perform additional tasks related to authentication.



Note

If you decide to configure plain text authentication or no authentication, see the **authentication** command provided in *OSPF Commands on Cisco IOS XR Software* module in *Routing Command Reference for Cisco CRS Routers*.

SUMMARY STEPS

1. Do one of the following:
 - **show ospf** [*process-name*]
 - **show ospfv3** [*process-name*]
2. **configure**
3. Do one of the following:
 - **router ospf** *process-name*
 - **router ospfv3** *process-name*
4. **router-id** { *router-id* }
5. **area** *area-id*
6. **virtual-link** *router-id*
7. **authentication message-digest**
8. **message-digest-key** *key-id* **md5** { *key* | **clear** *key* | **encrypted** *key* }
9. Repeat all of the steps in this task on the ABR that is at the other end of the virtual link. Specify the same key ID and key that you specified for the virtual link on this router.
10. **commit**
11. Do one of the following:
 - **show ospf** [*process-name*] [*area-id*] **virtual-links**
 - **show ospfv3** [*process-name*] **virtual-links**

DETAILED STEPS

	Command or Action	Purpose
Step 1	Do one of the following: <ul style="list-style-type: none"> • show ospf [<i>process-name</i>] • show ospfv3 [<i>process-name</i>] Example: <pre>RP/0/RP0/CPU0:router# show ospf</pre> or <pre>RP/0/RP0/CPU0:router# show ospfv3</pre>	(Optional) Displays general information about OSPF routing processes. <ul style="list-style-type: none"> • The output displays the router ID of the local router. You need this router ID to configure the other end of the link.
Step 2	configure	
Step 3	Do one of the following: <ul style="list-style-type: none"> • router ospf <i>process-name</i> • router ospfv3 <i>process-name</i> Example: <pre>RP/0/RP0/CPU0:router(config)# router ospf 1</pre> or <pre>RP/0/RP0/CPU0:router(config)# router ospfv3 1</pre>	Enables OSPF routing for the specified routing process and places the router in router configuration mode. or Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 4	router-id { <i>router-id</i> } Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3</pre>	Configures a router ID for the OSPF process. Note We recommend using a stable IPv4 address as the router ID.
Step 5	area <i>area-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# area 1</pre>	Enters area configuration mode and configures a nonbackbone area for the OSPF process. <ul style="list-style-type: none"> • The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 6	virtual-link <i>router-id</i> Example: <pre>RRP/0/RP0/CPU0:router(config-ospf-ar)# virtual-link 10.3.4.5</pre>	Defines an OSPF virtual link. <ul style="list-style-type: none"> • See .
Step 7	authentication message-digest Example:	Selects MD5 authentication for this virtual link.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router (config-ospf-ar-vl)#authentication message-digest	
Step 8	<p>message-digest-key <i>key-id</i> md5 { <i>key</i> clear <i>key</i> encrypted <i>key</i> }</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router (config-ospf-ar-vl)#message-digest-key 4 md5 yourkey</pre>	<p>Defines an OSPF virtual link.</p> <ul style="list-style-type: none"> • See to understand a virtual link. • The <i>key-id</i> argument is a number in the range from 1 to 255. The <i>key</i> argument is an alphanumeric string of up to 16 characters. The routers at both ends of the virtual link must have the same key identifier and key to be able to route OSPF traffic. • The authentication-key <i>key</i> command is not supported for OSPFv3. • Once the key is encrypted it must remain encrypted.
Step 9	Repeat all of the steps in this task on the ABR that is at the other end of the virtual link. Specify the same key ID and key that you specified for the virtual link on this router.	—
Step 10	commit	
Step 11	<p>Do one of the following:</p> <ul style="list-style-type: none"> • show ospf [<i>process-name</i>] [<i>area-id</i>] virtual-links • show ospfv3 [<i>process-name</i>] virtual-links <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show ospf 1 2 virtual-links or RP/0/RP0/CPU0:router# show ospfv3 1 virtual-links</pre>	(Optional) Displays the parameters and the current state of OSPF virtual links.

Examples

In the following example, the **show ospfv3 virtual links** EXEC configuration command verifies that the OSPF_VL0 virtual link to the OSPFv3 neighbor is up, the ID of the virtual link interface is 2, and the IPv6 address of the virtual link endpoint is 2003:3000::1.

```
show ospfv3 virtual-links

Virtual Links for OSPFv3 1

Virtual Link OSPF_VL0 to router 10.0.0.3 is up
Interface ID 2, IPv6 address 2003:3000::1
Run as demand circuit
DoNotAge LSA allowed.
Transit area 0.1.20.255, via interface GigabitEthernet 0/1/0/1, Cost of using 2
Transmit Delay is 5 sec, State POINT_TO_POINT,
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:02
```

```

Adjacency State FULL (Hello suppressed)
Index 0/2/3, retransmission queue length 0, number of retransmission 1
First 0(0)/0(0)/0(0) Next 0(0)/0(0)/0(0)
Last retransmission scan length is 1, maximum is 1
Last retransmission scan time is 0 msec, maximum is 0 msec

```

Check for lines:

```

Virtual Link OSPF_VL0 to router 10.0.0.3 is up
Adjacency State FULL (Hello suppressed)

```

State is up and Adjacency State is FULL

Summarizing Subnetwork LSAs on an OSPF ABR

If you configured two or more subnetworks when you assigned your IP addresses to your interfaces, you might want the software to summarize (aggregate) into a single LSA all of the subnetworks that the local area advertises to another area. Such summarization would reduce the number of LSAs and thereby conserve network resources. This summarization is known as interarea route summarization. It applies to routes from within the autonomous system. It does not apply to external routes injected into OSPF by way of redistribution.

This task configures OSPF to summarize subnetworks into one LSA, by specifying that all subnetworks that fall into a range are advertised together. This task is performed on an ABR only.

SUMMARY STEPS

1. **configure**
2. Do one of the following:
 - **router ospf** *process-name*
 - **router ospfv3** *process-name*
3. **router-id** { *router-id* }
4. **area** *area-id*
5. Do one of the following:
 - **range** *ip-address mask* [**advertise** | **not-advertise**]
 - **range** *ipv6-prefix / prefix-length* [**advertise** | **not-advertise**]
6. **interface** *type interface-path-id*
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	Do one of the following: <ul style="list-style-type: none"> • router ospf <i>process-name</i> • router ospfv3 <i>process-name</i> Example: <pre>RP/0/RP0/CPU0:router(config)# router ospf 1</pre>	Enables OSPF routing for the specified routing process and places the router in router configuration mode. or Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.

	Command or Action	Purpose
	or <pre>RP/0/RP0/CPU0:router(config)# router ospfv3 1</pre>	Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	router-id { <i>router-id</i> } Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3</pre>	Configures a router ID for the OSPF process. Note We recommend using a stable IPv4 address as the router ID.
Step 4	area <i>area-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# area 0</pre>	Enters area configuration mode and configures a nonbackbone area for the OSPF process. <ul style="list-style-type: none"> The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 5	Do one of the following: <ul style="list-style-type: none"> range <i>ip-address mask</i> [advertise not-advertise] range <i>ipv6-prefix / prefix-length</i> [advertise not-advertise] Example: <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# range 192.168.0.0 255.255.0.0 advertise</pre> or <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# range 4004:f000::/32 advertise</pre>	Consolidates and summarizes OSPF routes at an area boundary. <ul style="list-style-type: none"> The advertise keyword causes the software to advertise the address range of subnetworks in a Type 3 summary LSA. The not-advertise keyword causes the software to suppress the Type 3 summary LSA, and the subnetworks in the range remain hidden from other areas. In the first example, all subnetworks for network 192.168.0.0 are summarized and advertised by the ABR into areas outside the backbone. In the second example, two or more IPv4 interfaces are covered by a 192.x.x network.
Step 6	interface <i>type interface-path-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/2/0/3</pre>	Enters interface configuration mode and associates one or more interfaces to the area.
Step 7	commit	

Redistribute Routes into OSPF

This task redistributes routes from an IGP (could be a different OSPF process) into OSPF.

Before you begin

For information about configuring routing policy, see *Implementing Routing Policy on Cisco IOS XR Software* module in the *Routing Configuration Guide for Cisco CRS Routers*.

SUMMARY STEPS

1. **configure**
2. Do one of the following:
 - **router ospf** *process-name*
 - **router ospfv3** *process-name*
3. **router-id** { *router-id* }
4. **redistribute** *protocol* [*process-id*] { **level-1** | **level-1-2** | **level-2** } [**metric** *metric-value*] [**metric-type** *type-value*] [**match** { **external** [**1** | **2**] } [**tag** *tag-value*] [**route-policy** *policy-name*]
5. Do one of the following:
 - **summary-prefix** *address mask* [**not-advertise**] [**tag** *tag*]
 - **summary-prefix** *ipv6-prefix / prefix-length* [**not-advertise**] [**tag** *tag*]
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	Do one of the following: <ul style="list-style-type: none"> • router ospf <i>process-name</i> • router ospfv3 <i>process-name</i> Example: <pre>RP/0/RP0/CPU0:router(config)# router ospf 1</pre> or <pre>RP/0/RP0/CPU0:router(config)# router ospfv3 1</pre>	Enables OSPF routing for the specified routing process and places the router in router configuration mode. or Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	router-id { <i>router-id</i> } Example: <pre>RRP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3</pre>	Configures a router ID for the OSPF process. Note We recommend using a stable IPv4 address as the router ID.
Step 4	redistribute <i>protocol</i> [<i>process-id</i>] { level-1 level-1-2 level-2 } [metric <i>metric-value</i>] [metric-type <i>type-value</i>] [match { external [1 2] } [tag <i>tag-value</i>] [route-policy <i>policy-name</i>] Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# redistribute bgp 100</pre>	Redistributes OSPF routes from one routing domain to another routing domain. or Redistributes OSPFv3 routes from one routing domain to another routing domain. <ul style="list-style-type: none"> • This command causes the router to become an ASBR by definition.

	Command or Action	Purpose
	<p>or</p> <pre>RP/0/RP0/CPU0:router(config-router)#redistribute bgp 110</pre>	<ul style="list-style-type: none"> • OSPF tags all routes learned through redistribution as external. • The protocol and its process ID, if it has one, indicate the protocol being redistributed into OSPF. • The metric is the cost you assign to the external route. The default is 20 for all protocols except BGP, whose default metric is 1. • The OSPF example redistributes BGP autonomous system 1, Level 1 routes into OSPF as Type 2 external routes. • The OSPFv3 example redistributes BGP autonomous system 1, Level 1 and 2 routes into OSPF. The external link type associated with the default route advertised into the OSPFv3 routing domain is the Type 1 external route. <p>Note RPL is not supported for OSPFv3.</p>
Step 5	<p>Do one of the following:</p> <ul style="list-style-type: none"> • summary-prefix <i>address mask</i> [not-advertise] [tag tag] • summary-prefix <i>ipv6-prefix / prefix-length</i> [not-advertise] [tag tag] <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# summary-prefix 10.1.0.0 255.255.0.0</pre> <p>or</p> <pre>RP/0/RP0/CPU0:router(config-router)# summary-prefix 2010:11:22::/32</pre>	<p>(Optional) Creates aggregate addresses for OSPF.</p> <p>or</p> <p>(Optional) Creates aggregate addresses for OSPFv3.</p> <ul style="list-style-type: none"> • This command provides external route summarization of the non-OSPF routes. • External ranges that are being summarized should be contiguous. Summarization of overlapping ranges from two different routers could cause packets to be sent to the wrong destination. • This command is optional. If you do not specify it, each route is included in the link-state database and advertised in LSAs. • In the OSPFv2 example, the summary address 10.1.0.0 includes address 10.1.1.0, 10.1.2.0, 10.1.3.0, and so on. Only the address 10.1.0.0 is advertised in an external LSA. • In the OSPFv3 example, the summary address 2010:11:22::/32 has addresses such as 2010:11:22:0:1000::1, 2010:11:22:0:2000:679:1, and so on. Only the address 2010:11:22::/32 is advertised in the external LSA.
Step 6	commit	

Configuring OSPF Shortest Path First Throttling

This task explains how to configure SPF scheduling in millisecond intervals and potentially delay SPF calculations during times of network instability. This task is optional.

SUMMARY STEPS

1. **configure**
2. Do one of the following:
 - **router ospf** *process-name*
 - **router ospfv3** *process-name*
3. **router-id** { *router-id* }
4. **timers throttle spf** *spf-start spf-hold spf-max-wait*
5. **area** *area-id*
6. **interface** *type interface-path-id*
7. **commit**
8. Do one of the following:
 - **show ospf** [*process-name*]
 - **show ospfv3** [*process-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	Do one of the following: <ul style="list-style-type: none"> • router ospf <i>process-name</i> • router ospfv3 <i>process-name</i> Example: <pre>RP/0/RP0/CPU0:router(config)# router ospf 1</pre> or <pre>RP/0/RP0/CPU0:router(config)# router ospfv3 1</pre>	Enables OSPF routing for the specified routing process and places the router in router configuration mode. or Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	router-id { <i>router-id</i> } Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3</pre>	Configures a router ID for the OSPF process. Note We recommend using a stable IPv4 address as the router ID.
Step 4	timers throttle spf <i>spf-start spf-hold spf-max-wait</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# timers throttle spf 10 4800 90000</pre>	Sets SPF throttling timers.

	Command or Action	Purpose
Step 5	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config-ospf)# area 0	Enters area configuration mode and configures a backbone area. <ul style="list-style-type: none"> The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 6	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/1/0/3	Enters interface configuration mode and associates one or more interfaces to the area.
Step 7	commit	
Step 8	Do one of the following: <ul style="list-style-type: none"> show ospf [<i>process-name</i>] show ospfv3 [<i>process-name</i>] Example: RP/0/RP0/CPU0:router# show ospf 1 OR RP/0/RP0/CPU0:router# RP/0/RP0/CPU0:router# show ospfv3 2	(Optional) Displays SPF throttling timers.

Examples

In the following example, the **show ospf** EXEC configuration command is used to verify that the initial SPF schedule delay time, minimum hold time, and maximum wait time are configured correctly. Additional details are displayed about the OSPF process, such as the router type and redistribution of routes.

```
show ospf 1
```

```
Routing Process "ospf 1" with ID 192.168.4.3
  Supports only single TOS(TOS0) routes
  Supports opaque LSA
  It is an autonomous system boundary router
  Redistributing External Routes from,
    ospf 2
  Initial SPF schedule delay 5 msec
  Minimum hold time between two consecutive SPF's 100 msec
  Maximum wait time between two consecutive SPF's 1000 msec
  Minimum LSA interval 5 secs. Minimum LSA arrival 1 secs
  Number of external LSA 0. Checksum Sum 00000000
  Number of opaque AS LSA 0. Checksum Sum 00000000
  Number of DCbitless external and opaque AS LSA 0
  Number of DoNotAge external and opaque AS LSA 0
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  External flood list length 0
```



```
Non-Stop Forwarding enabled
```



Note For a description of each output display field, see the **show ospf** command in the *OSPF Commands on Cisco IOS XR Software* module in *Routing Command Reference for Cisco CRS Routers*.

Configuring Nonstop Forwarding Specific to Cisco for OSPF Version 2

This task explains how to configure OSPF NSF specific to Cisco on your NSF-capable router. This task is optional.

Before you begin

OSPF NSF requires that all neighbor networking devices be NSF aware, which happens automatically after you install the Cisco IOS XR software image on the router. If an NSF-capable router discovers that it has non-NSF-aware neighbors on a particular network segment, it disables NSF capabilities for that segment. Other network segments composed entirely of NSF-capable or NSF-aware routers continue to provide NSF capabilities.



Note The following are restrictions when configuring nonstop forwarding:

- OSPF Cisco NSF for virtual links is not supported.
- Neighbors must be NSF aware.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **router-id** { *router-id* }
4. Do one of the following:
 - **nsf cisco**
 - **nsf cisco enforce global**
5. **nsf interval** *seconds*
6. **nsfflush-delay-time***seconds*
7. **nsflifetime***seconds*
8. **nsfietf**
9. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	

	Command or Action	Purpose
Step 2	<p>router ospf <i>process-name</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config)# router ospf 1</pre>	<p>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</p> <p>Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.</p>
Step 3	<p>router-id { <i>router-id</i> }</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3</pre>	<p>Configures a router ID for the OSPF process.</p> <p>Note We recommend using a stable IPv4 address as the router ID.</p>
Step 4	<p>Do one of the following:</p> <ul style="list-style-type: none"> • nsf cisco • nsf cisco enforce global <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# nsf cisco enforce global</pre>	<p>Enables Cisco NSF operations for the OSPF process.</p> <ul style="list-style-type: none"> • Use the nsf cisco command without the optional enforce and global keywords to abort the NSF restart mechanism on the interfaces of detected non-NSF neighbors and allow NSF neighbors to function properly. • Use the nsf cisco command with the optional enforce and global keywords if the router is expected to perform NSF during restart. However, if non-NSF neighbors are detected, NSF restart is canceled for the entire OSPF process.
Step 5	<p>nsf interval <i>seconds</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# nsf interval 120</pre>	<p>Sets the minimum time between NSF restart attempts.</p> <p>Note When you use this command, the OSPF process must be up for at least 90 seconds before OSPF attempts to perform an NSF restart.</p>
Step 6	<p>nsfflush-delay-time <i>seconds</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# nsf flush-delay-time 1000</pre>	<p>Sets the maximum time allowed for external route learning in seconds.</p>
Step 7	<p>nsflifetime <i>seconds</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# nsf lifetime 90</pre>	<p>Sets the maximum route lifetime of NSF following a restart in seconds.</p>
Step 8	<p>nsfietf</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# nsf ietf</pre>	<p>Enables ietf graceful restart.</p>
Step 9	commit	

Configuring OSPF Version 2 for MPLS Traffic Engineering

This task explains how to configure OSPF for MPLS TE. This task is optional.

For a description of the MPLS TE tasks and commands that allow you to configure the router to support tunnels, configure an MPLS tunnel that OSPF can use, and troubleshoot MPLS TE, see *Implementing MPLS Traffic Engineering* on *Cisco IOS XR Software* module of the *MPLS Configuration Guide for the Cisco CRS Routers*

Before you begin

Your network must support the following features before you enable MPLS TE for OSPF on your router:

- MPLS
- IP Cisco Express Forwarding (CEF)



Note You must enter the commands in the following task on every OSPF router in the traffic-engineered portion of your network.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **router-id** { *router-id* }
4. **mpls traffic-eng router-id** *interface-type interface-instance*
5. **area** *area-id*
6. **mpls traffic-eng**
7. **interface** *type interface-path-id*
8. **commit**
9. **show ospf** [*process-name*] [*area-id*] **mpls traffic-eng** { **link** | **fragment** }

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospf 1	Enables OSPF routing for the specified routing process and places the router in router configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	router-id { <i>router-id</i> } Example: RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3	Configures a router ID for the OSPF process. Note We recommend using a stable IPv4 address as the router ID.

	Command or Action	Purpose
Step 4	mpls traffic-eng router-id <i>interface-type</i> <i>interface-instance</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# mpls traffic-eng router-id loopback 0</pre>	(Optional) Specifies that the traffic engineering router identifier for the node is the IP address associated with a given interface. <ul style="list-style-type: none"> • This IP address is flooded to all nodes in TE LSAs. • For all traffic engineering tunnels originating at other nodes and ending at this node, you must set the tunnel destination to the traffic engineering router identifier of the destination node because that is the address that the traffic engineering topology database at the tunnel head uses for its path calculation. • We recommend that loopback interfaces be used for MPLS TE router ID because they are more stable than physical interfaces.
Step 5	area <i>area-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# area 0</pre>	Enters area configuration mode and configures an area for the OSPF process. <ul style="list-style-type: none"> • The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area.
Step 6	mpls traffic-eng Example: <pre>RP/0/RP0/CPU0:router(config-ospf)# mpls traffic-eng</pre>	Configures the MPLS TE under the OSPF area.
Step 7	interface <i>type interface-path-id</i> Example: <pre>RP/0/RP0/CPU0:router(config-ospf-ar)# interface interface loopback0</pre>	Enters interface configuration mode and associates one or more interfaces to the area.
Step 8	commit	
Step 9	show ospf [<i>process-name</i>] [<i>area-id</i>] mpls traffic-eng { link fragment } Example: <pre>RP/0/RP0/CPU0:router# show ospf 1 0 mpls traffic-eng link</pre>	(Optional) Displays information about the links and fragments available on the local router for MPLS TE.

Examples

This section provides the following output examples:

Sample Output for the show ospf Command Before Configuring MPLS TE

In the following example, the **show route ospf EXEC** configuration command verifies that GigabitEthernet interface 0/3/0/0 exists and MPLS TE is not configured:

```
show route ospf 1

O   11.0.0.0/24 [110/15] via 0.0.0.0, 3d19h, tunnel-te1
O   192.168.0.12/32 [110/11] via 11.1.0.2, 3d19h, GigabitEthernet0/3/0/0
O   192.168.0.13/32 [110/6] via 0.0.0.0, 3d19h, tunnel-te1
```

Sample Output for the show ospf mpls traffic-eng Command

In the following example, the **show ospf mpls traffic-eng EXEC** configuration command verifies that the MPLS TE fragments are configured correctly:

```
show ospf 1 mpls traffic-eng fragment

OSPF Router with ID (192.168.4.3) (Process ID 1)

Area 0 has 1 MPLS TE fragment. Area instance is 3.
MPLS router address is 192.168.4.2
Next fragment ID is 1

Fragment 0 has 1 link. Fragment instance is 3.
Fragment has 0 link the same as last update.
Fragment advertise MPLS router address
  Link is associated with fragment 0. Link instance is 3
  Link connected to Point-to-Point network
  Link ID :55.55.55.55
  Interface Address :192.168.50.21
  Neighbor Address :192.168.4.1
  Admin Metric :0
  Maximum bandwidth :19440000
  Maximum global pool reservable bandwidth :25000000
  Maximum sub pool reservable bandwidth   :3125000
  Number of Priority :8
  Global pool unreserved BW
  Priority 0 : 25000000 Priority 1 : 25000000
  Priority 2 : 25000000 Priority 3 : 25000000
  Priority 4 : 25000000 Priority 5 : 25000000
  Priority 6 : 25000000 Priority 7 : 25000000
  Sub pool unreserved BW
  Priority 0 : 3125000 Priority 1 : 3125000
  Priority 2 : 3125000 Priority 3 : 3125000
  Priority 4 : 3125000 Priority 5 : 3125000
  Priority 6 : 3125000 Priority 7 : 3125000
  Affinity Bit :0
```

In the following example, the **show ospf mpls traffic-eng EXEC** configuration command verifies that the MPLS TE links on area instance 3 are configured correctly:

```
show ospf mpls traffic-eng link

OSPF Router with ID (192.168.4.1) (Process ID 1)

Area 0 has 1 MPLS TE links. Area instance is 3.

Links in hash bucket 53.
```

```

Link is associated with fragment 0. Link instance is 3
Link connected to Point-to-Point network
Link ID :192.168.50.20
Interface Address :192.168.20.50
Neighbor Address :192.168.4.1
Admin Metric :0
Maximum bandwidth :19440000
Maximum global pool reservable bandwidth :25000000
Maximum sub pool reservable bandwidth :3125000
Number of Priority :8
Global pool unreserved BW
Priority 0 : 25000000 Priority 1 : 25000000
Priority 2 : 25000000 Priority 3 : 25000000
Priority 4 : 25000000 Priority 5 : 25000000
Priority 6 : 25000000 Priority 7 : 25000000
Sub pool unreserved BW
Priority 0 : 3125000 Priority 1 : 3125000
Priority 2 : 3125000 Priority 3 : 3125000
Priority 4 : 3125000 Priority 5 : 3125000
Priority 6 : 3125000 Priority 7 : 3125000
Affinity Bit :0

```

Sample Output for the show ospf Command After Configuring MPLS TE

In the following example, the `show route ospf EXEC` configuration command verifies that the MPLS TE tunnels replaced GigabitEthernet interface 0/3/0/0 and that configuration was performed correctly:

```

show route ospf 1

O E2 192.168.10.0/24 [110/20] via 0.0.0.0, 00:00:15, tunnel2
O E2 192.168.11.0/24 [110/20] via 0.0.0.0, 00:00:15, tunnel2
O E2 192.168.1244.0/24 [110/20] via 0.0.0.0, 00:00:15, tunnel2
O 192.168.12.0/24 [110/2] via 0.0.0.0, 00:00:15, tunnel2

```

Configuring OSPFv3 Graceful Restart

This task explains how to configure a graceful restart for an OSPFv3 process. This task is optional.

SUMMARY STEPS

1. **configure**
2. **router ospfv3** *process-name*
3. **graceful-restart**
4. **graceful-restart lifetime**
5. **graceful-restart interval** *seconds*
6. **graceful-restart helper disable**
7. **commit**
8. **show ospfv3** [*process-name* [*area-id*]] **database grace**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	

	Command or Action	Purpose
Step 2	router ospfv3 <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospfv3 test	Enters router configuration mode for OSPFv3. The process name is a WORD that uniquely identifies an OSPF routing process. The process name is any alphanumeric string no longer than 40 characters without spaces.
Step 3	graceful-restart Example: RP/0/RP0/CPU0:router(config-ospfv3)#graceful-restart	Enables graceful restart on the current router.
Step 4	graceful-restart lifetime Example: RP/0/RP0/CPU0:router(config-ospfv3)# graceful-restart lifetime 120	Specifies a maximum duration for a graceful restart. <ul style="list-style-type: none"> • The default lifetime is 95 seconds. • The range is 90 to 3600 seconds.
Step 5	graceful-restart interval <i>seconds</i> Example: RP/0/RP0/CPU0:router(config-ospfv3)# graceful-restart interval 120	Specifies the interval (minimal time) between graceful restarts on the current router. <ul style="list-style-type: none"> • The default value for the interval is 90 seconds. • The range is 90 to 3600 seconds.
Step 6	graceful-restart helper disable Example: RP/0/RP0/CPU0:router(config-ospfv3)# graceful-restart helper disable	Disables the helper capability.
Step 7	commit	
Step 8	show ospfv3 [<i>process-name</i> [<i>area-id</i>]] database grace Example: RP/0/RP0/CPU0:router# show ospfv3 1 database grace	Displays the state of the graceful restart link.

Displaying Information About Graceful Restart

This section describes the tasks you can use to display information about a graceful restart.

- To see if the feature is enabled and when the last graceful restart ran, use the **show ospf** command. To see details for an OSPFv3 instance, use the **show ospfv3** *process-name* [*area-id*] **database grace** command.

Displaying the State of the Graceful Restart Feature

The following screen output shows the state of the graceful restart capability on the local router:

```
RP/0/RP0/CPU0:router# show ospfv3 1
```

```

Routing Process "ospfv3 1" with ID 2.2.2.2
Initial SPF schedule delay 5000 msec
Minimum hold time between two consecutive SPF's 10000 msec
Maximum wait time between two consecutive SPF's 10000 msec
Initial LSA throttle delay 0 msec
Minimum hold time for LSA throttle 5000 msec
Maximum wait time for LSA throttle 5000 msec
Minimum LSA arrival 1000 msec
LSA group pacing timer 240 secs
Interface flood pacing timer 33 msec
Retransmission pacing timer 66 msec
Maximum number of configured interfaces 255
Number of external LSA 0. Checksum Sum 00000000
Number of areas in this router is 1. 1 normal 0 stub 0 nssa
Graceful Restart enabled, last GR 11:12:26 ago (took 6 secs)
  Area BACKBONE(0)
    Number of interfaces in this area is 1
    SPF algorithm executed 1 times
    Number of LSA 6. Checksum Sum 0x0268a7
    Number of DCbitless LSA 0
    Number of indication LSA 0
    Number of DoNotAge LSA 0
    Flood list length 0

```

Displaying Graceful Restart Information for an OSPFv3 Instance

The following screen output shows the link state for an OSPFv3 instance:

```

RP/0/RP0/CPU0:router# show ospfv3 1 database grace

      OSPFv3 Router with ID (1.1.1.1) (Process ID 1)

      Grace (Type-11) Link States (Area 0)

LS age: 2
LS Type: Grace Links
Link State ID: 34
Advertising Router: 1.1.1.1
LS Seq Number: 80000001
Checksum: 0x7a4a
Length: 36
  Grace Period : 90
  Graceful Restart Reason : Software reload/upgrade

```

Configuring an OSPFv2 Sham Link

This task explains how to configure a provider edge (PE) router to establish an OSPFv2 sham link connection across a VPN backbone. This task is optional.

Before you begin

Before configuring a sham link in a Multiprotocol Label Switching (MPLS) VPN between provider edge (PE) routers, OSPF must be enabled as follows:

- Create an OSPF routing process.

- Configure a loopback interface that belongs to VRF and assign a IPv4 address with the host mask to it.
- Configure the sham link under the area submode.

See [Enabling OSPF, on page 383](#) for information on these OSPF configuration prerequisites.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **vrf** *vrf-name*
4. **ipv4 address** *ip-address mask*
5. **end**
6. **router ospf** *instance-id*
7. **vrf** *vrf-name*
8. **router-id** { *router-id* }
9. **redistribute bgp** *process-id*
10. **area** *area-id*
11. **sham-link** *source-address destination-address*
12. **cost** *cost*
13. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config)# interface loopback 3	Enters interface configuration mode.
Step 3	vrf <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config-if)# vrf vrf1	Assigns an interface to the VPN routing and forwarding (VRF) instance.
Step 4	ipv4 address <i>ip-address mask</i> Example: RP/0/RP0/CPU0:router(config-if)# ipv4 address 172.18.189.38 255.255.255.225	Assigns an IP address and subnet mask to the interface.
Step 5	end Example: RP/0/RP0/CPU0:router(config-if)# end	Saves configuration changes. When you issue the end command, the system prompts you to commit changes: Uncommitted changes found, commit them before

	Command or Action	Purpose
		<p>exiting (yes/no/cancel) ?[cancel] :</p> <ul style="list-style-type: none"> • Entering yes saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering no exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
Step 6	<p>router ospf <i>instance-id</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config)# router ospf isp</pre>	Enables OSPF routing for the specified routing process, and places the router in router configuration mode. In this example, the OSPF instance is called isp.
Step 7	<p>vrf <i>vrf-name</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf)# vrf vrf1</pre>	Creates a VRF instance and enters VRF configuration mode.
Step 8	<p>router-id { <i>router-id</i> }</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-vrf)# router-id 192.168.4.3</pre>	<p>Configures a router ID for the OSPF process.</p> <p>Note We recommend using a stable IPv4 address as the router ID.</p>
Step 9	<p>redistribute bgp <i>process-id</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-vrf)# redistribute bgp 1</pre>	<p>Redistributes OSPF routes from the one routing domain to another routing domain.</p> <ul style="list-style-type: none"> • This command causes the router to become an ASBR by definition. • OSPF tags all routes learned through redistribution as external. • The protocol and its process ID, if it has one, indicate the protocol being redistributed into OSPF. • The BGP MED value is copied to the LSA metric field when BGP VPN routes are redistributed to OSPF.
Step 10	<p>area <i>area-id</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-ospf-vrf)# area 0</pre>	<p>Enters area configuration mode and configures an area for the OSPF process.</p> <ul style="list-style-type: none"> • The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area

	Command or Action	Purpose
		1000 or area 0.0.3.232. However, you must choose one form or the other for an area.
Step 11	sham-link <i>source-address destination-address</i> Example: RP/0/RP0/CPU0:router(config-ospf-vrf-ar)# sham-link 10.0.0.1 10.0.0.3	Configures a point-to-point unnumbered interface between two VPN sites.
Step 12	cost <i>cost</i> Example: RP/0/RP0/CPU0:router(config-ospf-vrf-ar-sl)# cost 76	Explicitly specifies the cost of sending a packet on an OSPF interface. The specified cost overrides the auto-costing calculated default value for interfaces.
Step 13	commit	

Enabling Nonstop Routing for OSPFv2

This optional task describes how to enable nonstop routing (NSR) for OSPFv2 process. NSR is disabled by default. When NSR is enabled, OSPF process on the active RP synchronizes all necessary data and states with the OSPF process on the standby RP. When the switchover happens, OSPF process on the newly active RP has all the necessary data and states to continue running and does not require any help from its neighbors.

Step 1 configure

Enter the global configuration mode.

Step 2 router ospf *instance-id*

Example:

```
RP/0/RP0/CPU0:router(config)# router ospf isp
```

Enable OSPF routing for the specified routing process. In this example, the OSPF instance is called isp.

Step 3 nsr

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# nsr
```

Enable NSR for the OSPFv2 process.

Step 4 commit

Commit your configuration.

Enabling Nonstop Routing for OSPFv3

This task describes how to enable nonstop routing (NSR) for OSPFv3 process. NSR is disabled by default. When NSR is enabled, OSPF process on the active RP synchronizes all necessary data and states with the OSPF process on the standby RP. When the switchover happens, OSPF process on the newly active RP has all the necessary data and states to continue running and does not require any help from its neighbors.

Step 1 **configure**

Enter the global configuration mode.

Step 2 **router ospfv3 *instance-id*****Example:**

```
RP/0/RP0/CPU0:router(config)# router ospfv3 isp
```

Enable OSPF routing for the specified routing process. In this example, the OSPF instance is called isp.

Step 3 **nsr****Example:**

```
RP/0/RP0/CPU0:router(config-ospfv3)# nsr
```

Enable NSR for the OSPFv3 process.

Step 4 **commit**

Commit your configuration.

Configuring OSPF SPF Prefix Prioritization

Perform this task to configure OSPF SPF (shortest path first) prefix prioritization.

SUMMARY STEPS

1. **configure**
2. **prefix-set *prefix-set name***
3. **route-policy *route-policy name* if destination in *prefix-set name* then set spf-priority {critical | high | medium} endif**
4. Use one of these commands:
 - **router ospf *ospf-name***
 - **router ospfv3 *ospfv3-name***
5. **spf prefix-priority route-policy *route-policy name***
6. **commit**
7. **show rpl route-policy *route-policy name* detail**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	prefix-set <i>prefix-set name</i> Example: <pre>RP/0/RP0/CPU0:router (config) #prefix-set ospf-critical-prefixes RP/0/RP0/CPU0:router (config-pfx) #66.0.0.0/16 RP/0/RP0/CPU0:router (config-pfx) #end-set</pre>	Configures the prefix set.
Step 3	route-policy <i>route-policy name</i> if destination in <i>prefix-set name</i> then set spf-priority {critical high medium} endif Example: <pre>RP/0/RP0/CPU0:router#route-policy ospf-spf-priority RP/0/RP0/CPU0:router (config-rpl) #if destination in ospf-critical-prefixes then set spf-priority critical endif RP/0/RP0/CPU0:router (config-rpl) #end-policy</pre>	Configures route policy and sets OSPF SPF priority.
Step 4	Use one of these commands: <ul style="list-style-type: none"> • router ospf <i>ospf-name</i> • router ospfv3 <i>ospfv3-name</i> Example: <pre>RP/0/RP0/CPU0:router# router ospf 1</pre> Or <pre>RP/0/RP0/CPU0:router# router ospfv3 1</pre>	Enters Router OSPF configuration mode.
Step 5	spf prefix-priority route-policy <i>route-policy name</i> Example: Or <pre>RP/0/RP0/CPU0:router (config-ospfv3) #spf prefix-priority route-policy ospf3-spf-priority</pre>	Configures SPF prefix-priority for the defined route policy. Note Configure the spf prefix-priority command under router OSPF.
Step 6	commit	
Step 7	show rpl route-policy <i>route-policy name</i> detail Example: <pre>RP/0/RP0/CPU0:router#show rpl route-policy ospf-spf-priority detail prefix-set ospf-critical-prefixes 66.0.0.0/16 end-set</pre>	Displays the set SPF prefix priority.

	Command or Action	Purpose
	<pre>! route-policy ospf-spf-priority if destination in ospf-critical-prefixes then set spf-priority critical endif end-policy !</pre>	

Enabling Multicast-intact for OSPFv2

This optional task describes how to enable multicast-intact for OSPFv2 routes that use IPv4 addresses.

SUMMARY STEPS

1. **configure**
2. **router ospf** *instance-id*
3. **mpls traffic-eng multicast-intact**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>instance-id</i> Example: RP/0/RP0/CPU0:router(config)# router ospf isp	Enables OSPF routing for the specified routing process, and places the router in router configuration mode. In this example, the OSPF instance is called isp.
Step 3	mpls traffic-eng multicast-intact Example: RP/0/RP0/CPU0:router(config-ospf)# mpls traffic-eng multicast-intact	Enables multicast-intact.
Step 4	commit	

Associating Interfaces to a VRF

This task explains how to associate an interface with a VPN Routing and Forwarding (VRF) instance.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **vrf** *vrf-name*
4. **area** *area-id*

5. `interface type interface-path-id`
6. `commit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospf 1	Enables OSPF routing for the specified routing process and places the router in router configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	vrf <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config-ospf)# vrf vrf1	Creates a VRF instance and enters VRF configuration mode.
Step 4	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-vrf)# area 0	Enters area configuration mode and configures an area for the OSPF process. <ul style="list-style-type: none"> • The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area.
Step 5	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-vrf-ar)# interface GigabitEthernet 0/0/0/0	Enters interface configuration mode and associates one or more interfaces to the VRF.
Step 6	<code>commit</code>	

Configuring OSPF as a Provider Edge to Customer Edge (PE-CE) Protocol

SUMMARY STEPS

1. `configure`
2. `router ospf process-name`
3. `vrf vrf-name`
4. `router-id {router-id}`
5. `redistribute protocol [process-id] {level-1 | level-1-2 | level-2} [metric metric-value] [metric-type type-value] [match {external [1 | 2]}] [tag tag-value] route-policy policy-name`
6. `area area-id`
7. `interface type interface-path-id`
8. `exit`
9. `domain-id [secondary] type {0005 | 0105 | 0205 | 8005} value value`

10. `domain-tag tag`
11. `disable-dn-bit-check`
12. `commit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospf 1	Enables OSPF routing for the specified routing process and places the router in router configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	vrf <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config-ospf)# vrf vrf1	Creates a VRF instance and enters VRF configuration mode.
Step 4	router-id { <i>router-id</i> } Example: RP/0/RP0/CPU0:router(config-ospf-vrf)# router-id 192.168.4.3	Configures a router ID for the OSPF process. Note We recommend using a stable IPv4 address as the router ID.
Step 5	redistribute <i>protocol</i> [<i>process-id</i>] { level-1 level-1-2 level-2 } [metric <i>metric-value</i>] [metric-type <i>type-value</i>] [match { external [1 2] }] [tag <i>tag-value</i>] route-policy <i>policy-name</i> Example: RP/0/RP0/CPU0:router(config-ospf-vrf)# redistribute bgp 1 level-1	Redistributes OSPF routes from one routing domain to another routing domain. <ul style="list-style-type: none"> • This command causes the router to become an ASBR by definition. • OSPF tags all routes learned through redistribution as external. • The protocol and its process ID, if it has one, indicate the protocol being redistributed into OSPF. • The metric is the cost you assign to the external route. The default is 20 for all protocols except BGP, whose default metric is 1. • The example shows the redistribution of BGP autonomous system 1, Level 1 routes into OSPF as Type 2 external routes.
Step 6	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-vrf)# area 0	Enters area configuration mode and configures an area for the OSPF process. <ul style="list-style-type: none"> • The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area

	Command or Action	Purpose
		1000 or area 0.0.3.232. However, you must choose one form or the other for an area.
Step 7	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-vrf)# interface GigabitEthernet 0/0/0/0	Enters interface configuration mode and associates one or more interfaces to the VRF.
Step 8	exit Example: RP/0/RP0/CPU0:router(config-if)# exit	Exits interface configuration mode.
Step 9	domain-id [<i>secondary</i>] type { 0005 0105 0205 8005 } value <i>value</i> Example: RP/0/RP0/CPU0:router(config-ospf-vrf)# domain-id type 0105 value 1AF234	Specifies the OSPF VRF domain ID. <ul style="list-style-type: none"> • The <i>value</i> argument is a six-octet hex number.
Step 10	domain-tag <i>tag</i> Example: RP/0/RP0/CPU0:router(config-ospf-vrf)# domain-tag 234	Specifies the OSPF VRF domain tag. <ul style="list-style-type: none"> • The valid range for <i>tag</i> is 0 to 4294967295.
Step 11	disable-dn-bit-check Example: RP/0/RP0/CPU0:router(config-ospf-vrf)# disable-dn-bit-check	Specifies that down bits should be ignored.
Step 12	commit	

Creating Multiple OSPF Instances (OSPF Process and a VRF)

This task explains how to create multiple OSPF instances. In this case, the instances are a normal OSPF instance and a VRF instance.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **area** *area-id*
4. **interface** *type interface-path-id*
5. **exit**
6. **vrf** *vrf-name*

7. **area** *area-id*
8. **interface** *type interface-path-id*
9. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospf 1	Enables OSPF routing for the specified routing process and places the router in router configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config-ospf)# area 0	Enters area configuration mode and configures a backbone area. <ul style="list-style-type: none"> The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 4	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/1/0/3	Enters interface configuration mode and associates one or more interfaces to the area.
Step 5	exit Example: RP/0/RP0/CPU0:router(config-ospf-ar)# exit	Enters OSPF configuration mode.
Step 6	vrf <i>vrf-name</i> Example: RP/0/RP0/CPU0:router(config-ospf)# vrf vrf1	Creates a VRF instance and enters VRF configuration mode.
Step 7	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-vrf)# area 0	Enters area configuration mode and configures an area for a VRF instance under the OSPF process. <ul style="list-style-type: none"> The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area.
Step 8	interface <i>type interface-path-id</i> Example:	Enters interface configuration mode and associates one or more interfaces to the VRF.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-ospf-vrf)# interface GigabitEthernet 0/0/0/0	
Step 9	commit	

Configuring Multi-area Adjacency

This task explains how to create multiple areas on an OSPF primary interface.

Before you begin



Note You can configure multi-area adjacency on any interface where only two OSPF speakers are attached. In the case of native broadcast networks, the interface must be configured as an OSPF point-to-point type using the **network point-to-point** command to enable the interface for a multi-area adjacency.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **area** *area-id*
4. **interface** *type interface-path-id*
5. **area** *area-id*
6. **multi-area-interface** *type interface-path-id*
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospf 1	Enables OSPF routing for the specified routing process and places the router in router configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config-ospf)# area 0	Enters area configuration mode and configures a backbone area. • The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

	Command or Action	Purpose
Step 4	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-ar)# interface Serial 0/1/0/3	Enters interface configuration mode and associates one or more interfaces to the area.
Step 5	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config-ospf)# area 1	Enters area configuration mode and configures an area used for multiple area adjacency. <ul style="list-style-type: none"> The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 6	multi-area-interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf)# multi-area-interface Serial 0/1/0/3	Enables multiple adjacencies for different OSPF areas and enters multi-area interface configuration mode.
Step 7	commit	

Configuring Label Distribution Protocol IGP Auto-configuration for OSPF

This task explains how to configure LDP auto-configuration for an OSPF instance.

Optionally, you can configure this feature for an area of an OSPF instance.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **mpls ldp auto-config**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospf 1	Enables OSPF routing for the specified routing process and places the router in router configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	mpls ldp auto-config Example:	Enables LDP IGP interface auto-configuration for an OSPF instance.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-ospf)# mpls ldp auto-config	<ul style="list-style-type: none"> Optionally, this command can be configured for an area of an OSPF instance.
Step 4	commit	

Configuring LDP IGP Synchronization: OSPF

Perform this task to configure LDP IGP Synchronization under OSPF.



Note By default, there is no synchronization between LDP and IGP.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. Use one of the following commands:
 - **mpls ldp sync**
 - **area** *area-id* **mpls ldp sync**
 - **area** *area-id* **interface** *name* **mpls ldp sync**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospf 100	Identifies the OSPF routing process and enters OSPF configuration mode.
Step 3	Use one of the following commands: <ul style="list-style-type: none"> • mpls ldp sync • area <i>area-id</i> mpls ldp sync • area <i>area-id</i> interface <i>name</i> mpls ldp sync Example: RP/0/RP0/CPU0:router(config-ospf)# mpls ldp sync	Enables LDP IGP synchronization on an interface.
Step 4	commit	

Configuring Authentication Message Digest Management for OSPF

This task explains how to manage authentication of a keychain on the OSPF interface.

Before you begin

A valid keychain must be configured before this task can be attempted.

To learn how to configure a keychain and its associated attributes, see the *Implementing Key Chain Management on Cisco IOS XR Software* module of the *System Security Configuration Guide for Cisco CRS Routers*.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **router-id** { *router-id* }
4. **area** *area-id*
5. **interface** *type interface-path-id*
6. **authentication** [**message-digest** *keychain* | **null**]
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospf 1	Enables OSPF routing for the specified routing process and places the router in router configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	router-id { <i>router-id</i> } Example: RP/0/RP0/CPU0:router(config-ospf)# router id 192.168.4.3	Configures a router ID for the OSPF process. Note We recommend using a stable IPv4 address as the router ID.
Step 4	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config-ospf)# area 1	Enters area configuration mode. The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 5	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet0/4/0/1	Enters interface configuration mode and associates one or more interfaces to the area.
Step 6	authentication [message-digest <i>keychain</i> null]	Configures an MD5 keychain.

	Command or Action	Purpose
	Example: RP/0/RP0/CPU0:router(config-ospf-ar-if)# authentication message-digest keychain ospf_int1	Note In the example, the <i>ospf_int1</i> keychain must be configured before you attempt this step.
Step 7	commit	

Examples

The following example shows how to configure the keychain *ospf_intf_1* that contains five key IDs. Each key ID is configured with different **send-lifetime** values; however, all key IDs specify the same text string for the key.

```
key chain ospf_intf_1
key 1
send-lifetime 11:30:30 May 1 2007 duration 600
cryptographic-algorithm MD5T
key-string clear ospf_intf_1
key 2
send-lifetime 11:40:30 May 1 2007 duration 600
cryptographic-algorithm MD5
key-string clear ospf_intf_1
key 3
send-lifetime 11:50:30 May 1 2007 duration 600
cryptographic-algorithm MD5
key-string clear ospf_intf_1
key 4
send-lifetime 12:00:30 May 1 2007 duration 600
cryptographic-algorithm MD5
key-string clear ospf_intf_1
key 5
send-lifetime 12:10:30 May 1 2007 duration 600
cryptographic-algorithm MD5
key-string clear ospf_intf_1
```

The following example shows that keychain authentication is enabled on the Gigabit Ethernet 0/4/0/1 interface:

```
show ospf 1 interface GigabitEthernet0/4/0/1
```

```
GigabitEthernet0/4/0/1 is up, line protocol is up
 Internet Address 100.10.10.2/24, Area 0
 Process ID 1, Router ID 2.2.2.1, Network Type BROADCAST, Cost: 1
 Transmit Delay is 1 sec, State DR, Priority 1
 Designated Router (ID) 2.2.2.1, Interface address 100.10.10.2
 Backup Designated router (ID) 1.1.1.1, Interface address 100.10.10.1
 Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
 Hello due in 00:00:02
 Index 3/3, flood queue length 0
 Next 0(0)/0(0)
 Last flood scan length is 2, maximum is 16
 Last flood scan time is 0 msec, maximum is 0 msec
 Neighbor Count is 1, Adjacent neighbor count is 1
 Adjacent with neighbor 1.1.1.1 (Backup Designated Router)
 Suppress hello for 0 neighbor(s)
 Keychain-based authentication enabled
 Key id used is 3
 Multi-area interface Count is 0
```

The following example shows output for configured keys that are active:

```
show key chain ospf_intf_1

Key-chain: ospf_intf_1/ -

Key 1 -- text "0700325C4836100B0314345D"
  cryptographic-algorithm -- MD5
  Send lifetime: 11:30:30, 01 May 2007 - (Duration) 600
  Accept lifetime: Not configured
Key 2 -- text "10411A0903281B051802157A"
  cryptographic-algorithm -- MD5
  Send lifetime: 11:40:30, 01 May 2007 - (Duration) 600
  Accept lifetime: Not configured
Key 3 -- text "06091C314A71001711112D5A"
  cryptographic-algorithm -- MD5
  Send lifetime: 11:50:30, 01 May 2007 - (Duration) 600 [Valid now]
  Accept lifetime: Not configured
Key 4 -- text "151D181C0215222A3C350A73"
  cryptographic-algorithm -- MD5
  Send lifetime: 12:00:30, 01 May 2007 - (Duration) 600
  Accept lifetime: Not configured
Key 5 -- text "151D181C0215222A3C350A73"
  cryptographic-algorithm -- MD5
  Send lifetime: 12:10:30, 01 May 2007 - (Duration) 600
  Accept lifetime: Not configured
```

Configuring Generalized TTL Security Mechanism (GTSM) for OSPF

This task explains how to set the security time-to-live mechanism on an interface for GTSM.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **router-id** { *router-id* }
4. **log adjacency changes** [*detail* | *disable*]
5. **nsf** { *cisco* [*enforce global*] | *ietf* [*helper disable*] }
6. **timers throttle spf** *spf-start* *spf-hold* *spf-max-wait*
7. **area** *area-id*
8. **interface** *type interface-path-id*
9. **security ttl** [*disable* | *hops* *hop-count*]
10. **commit**
11. **show ospf** [*process-name*] [*vrf* *vrf-name*] [*area-id*] **interface** [*type interface-path-id*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>process-name</i> Example:	Enables OSPF routing for the specified routing process and places the router in router configuration mode.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config)# router ospf 1	Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	router-id { <i>router-id</i> } Example: RP/0/RP0/CPU0:router(config-ospf)# router id 10.10.10.100	Configures a router ID for the OSPF process. Note We recommend using a stable IPv4 address as the router ID.
Step 4	log adjacency changes [detail disable] Example: RP/0/RP0/CPU0:router(config-ospf-ar-if)# log adjacency changes detail	(Optional) Requests notification of neighbor changes. <ul style="list-style-type: none"> • By default, this feature is enabled. • The messages generated by neighbor changes are considered notifications, which are categorized as severity Level 5 in the logging console command. The logging console command controls which severity level of messages are sent to the console. By default, all severity level messages are sent.
Step 5	nsf { cisco [enforce global] ietf [helper disable] } Example: RP/0/RP0/CPU0:router(config-ospf)# nsf ietf	(Optional) Configures NSF OSPF protocol. The example enables graceful restart.
Step 6	timers throttle spf <i>spf-start spf-hold spf-max-wait</i> Example: RP/0/RP0/CPU0:router(config-ospf)# timers throttle spf 500 500 10000	(Optional) Sets SPF throttling timers.
Step 7	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config-ospf)# area 1	Enters area configuration mode. The <i>area-id</i> argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 8	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet0/5/0/0	Enters interface configuration mode and associates one or more interfaces to the area.
Step 9	security ttl [disable hops <i>hop-count</i>] Example:	Sets the security TTL value in the IP header for OSPF packets.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-ospf-ar-if)# security ttl hopes 2	
Step 10	commit	
Step 11	show ospf [<i>process-name</i>] [vrf <i>vrf-name</i>] [<i>area-id</i>] interface [<i>type interface-path-id</i>] Example: RP/0/RP0/CPU0:router# show ospf 1 interface GigabitEthernet0/5/0/0	Displays OSPF interface information.

Examples

The following is sample output that displays the GTSM security TTL value configured on an OSPF interface:

```
show ospf 1 interface GigabitEthernet0/5/0/0

GigabitEthernet0/5/0/0 is up, line protocol is up
  Internet Address 120.10.10.1/24, Area 0
  Process ID 1, Router ID 100.100.100.100, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State BDR, Priority 1
  TTL security enabled, hop count 2
  Designated Router (ID) 102.102.102.102, Interface address 120.10.10.3
  Backup Designated router (ID) 100.100.100.100, Interface address 120.10.10.1
  Flush timer for old DR LSA due in 00:02:36
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:05
  Index 1/1, flood queue length 0
  Next 0(0)/0(0)
  Last flood scan length is 1, maximum is 4
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 102.102.102.102 (Designated Router)
  Suppress hello for 0 neighbor(s)
  Multi-area interface Count is 0
```

Verifying OSPF Configuration and Operation

This task explains how to verify the configuration and operation of OSPF.

SUMMARY STEPS

1. **show** { **ospf** | **ospfv3** } [*process-name*]
2. **show** { **ospf** | **ospfv3** } [*process-name*] **border-routers** [*router-id*]
3. **show** { **ospf** | **ospfv3** } [*process-name*] **database**
4. **show** { **ospf** | **ospfv3** } [*process-name*] [*area-id*] **flood-list interface** *type interface-path-id*
5. **show** { **ospf** | **ospfv3** } [*process-name*] [**vrf** *vrf-name*] [*area-id*] **interface** [*type interface-path-id*]
6. **show** { **ospf** | **ospfv3** } [*process-name*] [*area-id*] **neighbor** [*type interface-path-id*] [*neighbor-id*] [**detail**]

7. **clear** { **ospf** | **ospfv3** } [*process-name*] **process**
8. **clear**{**ospf**|**ospfv3**[*process-name*] **redistribution**
9. **clear**{**ospf**|**ospfv3**[*process-name*] **routes**
10. **clear**{**ospf**|**ospfv3**[*process-name*] **vrf** [*vrf-name*|**all**] {**process** |**redistribution**|**routes**|**statistics** [**interface** *type interface-path-id*] **message-queue**|**neighbor** }
11. **clear** { **ospf** | **ospfv3** } [*process-name*] **statistics** [**neighbor** [*type interface-path-id*]] [*ip-address*]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	show { ospf ospfv3 } [<i>process-name</i>] Example: RP/0/RP0/CPU0:router# show ospf group1	(Optional) Displays general information about OSPF routing processes.
Step 2	show { ospf ospfv3 } [<i>process-name</i>] border-routers [<i>router-id</i>] Example: RP/0/RP0/CPU0:router# show ospf group1 border-routers	(Optional) Displays the internal OSPF routing table entries to an ABR and ASBR.
Step 3	show { ospf ospfv3 } [<i>process-name</i>] database Example: RP/0/RP0/CPU0:router# show ospf group2 database	(Optional) Displays the lists of information related to the OSPF database for a specific router. <ul style="list-style-type: none"> • The various forms of this command deliver information about different OSPF LSAs.
Step 4	show { ospf ospfv3 } [<i>process-name</i>] [<i>area-id</i>] flood-list interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router# show ospf 100 flood-list interface GigabitEthernet 0/3/0/0	(Optional) Displays a list of OSPF LSAs waiting to be flooded over an interface.
Step 5	show { ospf ospfv3 } [<i>process-name</i>] [vrf <i>vrf-name</i>] [<i>area-id</i>] interface [<i>type interface-path-id</i>] Example: RP/0/RP0/CPU0:router# show ospf 100 interface GigabitEthernet 0/3/0/0	(Optional) Displays OSPF interface information.
Step 6	show { ospf ospfv3 } [<i>process-name</i>] [<i>area-id</i>] neighbor [<i>type interface-path-id</i>] [<i>neighbor-id</i>] [detail] Example: RP/0/RP0/CPU0:router# show ospf 100 neighbor	(Optional) Displays OSPF neighbor information on an individual interface basis.

	Command or Action	Purpose
Step 7	clear { ospf ospfv3 } [process-name] process Example: RP/0/ RP0/CPU0:router# clear ospf 100 process	(Optional) Resets an OSPF router process without stopping and restarting it.
Step 8	clear {ospf ospfv3} [process-name] redistribution Example: RP/0/RP0/CPU0:router#clear ospf 100 redistribution	Clears OSPF route redistribution.
Step 9	clear {ospf ospfv3} [process-name] routes Example: RP/0/RP0/CPU0:router#clear ospf 100 routes	Clears OSPF route table.
Step 10	clear {ospf ospfv3} [process-name] vrf [vrf-name all] {process redistribution routes statistics [interface type interface-path-id message-queue neighbor]} Example: RP/0/RP0/CPU0:router#clear ospf 100 vrf vrf_1 process	Clears OSPF route table.
Step 11	clear { ospf ospfv3 } [process-name] statistics [neighbor [type interface-path-id] [ip-address]] Example: RP/0/RP0/CPU0:router# clear ospf 100 statistics	(Optional) Clears the OSPF statistics of neighbor state transitions.

Configuring OSPF Queue Tuning Parameters

The following procedures explain how to limit the number of continuous incoming events processed, how to set the maximum number of rate-limited link-state advertisements (LSAs) processed per run, how to limit the number of summary or external Type 3 to Type 7 link-state advertisements (LSAs) processed per shortest path first (SPF) run, and how to set the high watermark for incoming priority events.

SUMMARY STEPS

1. **configure**
2. **router ospf process-name**
3. **queue dispatch incoming count**
4. **queue dispatch rate-limited-lsa count**
5. **queue dispatch spf-lsa-limit count**
6. **queue limit { high | medium | low } count**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router# router ospf ospf1	Enables OSPF routing for the specified routing process and places the router in router configuration mode. Note The <i>process-name</i> argument is any alphanumeric string no longer than 40 characters.
Step 3	queue dispatch incoming <i>count</i> Example: RP/0/RP0/CPU0:router# queue dispatch incoming 30	Limits the number of continuous incoming events processed.
Step 4	queue dispatch rate-limited-lsa <i>count</i> Example: RP/0/RP0/CPU0:router# queue dispatch rate-limited-lsa 3000	Sets the maximum number of rate-limited link-state advertisements (LSAs) processed per run.
Step 5	queue dispatch spf-lsa-limit <i>count</i> Example: RP/0/RP0/CPU0:router# queue dispatch spf-lsa-limit 2000	Limits the number of summary or external Type 3 to Type 7 link-state advertisements (LSAs) processed per shortest path first (SPF) run.
Step 6	queue limit { high medium low } <i>count</i> Example: RP/0/RP0/CPU0:router# (config-ospf)# queue limit high 1000	Sets the high watermark for incoming priority events, use the queue limit in router configuration mode.

Configuring IP Fast Reroute Loop-free Alternate

This task describes how to enable the IP fast reroute (IPFRR) per-link loop-free alternate (LFA) computation to converge traffic flows around link failures.

To enable protection on broadcast links, IPFRR and bidirectional forwarding detection (BFD) must be enabled on the interface under OSPF.

Enabling IPFRR LFA

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*

3. `area area-id`
4. `interface type interface-path-id`
5. `fast-reroute per-link { enable | disable }`
6. `commit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospf	Enables OSPF routing for the specified routing process and places the router in router configuration mode.
Step 3	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config-ospf)#area 1	Enters area configuration mode.
Step 4	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet0/5/0/0	Enters interface configuration mode and associates one or more interfaces to the area. .
Step 5	fast-reroute per-link { enable disable } Example: RP/0/RP0/CPU0:router(config-ospf-ar)#fast-reroute per-link enable	Enables or disables per-link LFA computation for the interface.
Step 6	<code>commit</code>	

Excluding an Interface From IP Fast Reroute Per-link Computation

SUMMARY STEPS

1. `configure`
2. `router ospf process-name`
3. `area area-id`
4. `interface type interface-path-id`
5. `fast-reroute per-link exclude interface type interface-path-id`
6. `commit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>process-name</i> Example: RP/0/RP0/CPU0:router(config)# router ospf	Enables the OSPF routing for the specified routing process and places the router in router configuration mode.
Step 3	area <i>area-id</i> Example: RP/0/RP0/CPU0:router(config)#area area-id	Enters area configuration mode.
Step 4	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf)#interface type interface-path-id	Enters interface configuration mode and associates one or more interfaces to the area.
Step 5	fast-reroute per-link exclude interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-ospf-ar)# fast-reroute per-link exclude interface GigabitEthernet0/5/0/1	Excludes an interface from IP fast reroute per-link computation.
Step 6	commit	

Enabling OSPF Interaction with SRMS Server

To enable OSPF interaction with SRMS server:

SUMMARY STEPS

1. **configure**
2. **router ospf** *instance-id*
3. **segment-routing mpls**
4. **segment-routing forwarding mpls**
5. **segment-routing prefix-sid-mapadvertise-local**
6. **segment-routing sr-preferprefix-list***[acl-name]*

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router ospf <i>instance-id</i> Example: RP/0/RP0/CPU0:router(config)# router ospf isp	Enables OSPF routing for the specified routing instance, and places the router in router configuration mode.

	Command or Action	Purpose
Step 3	segment-routing mpls Example: RP/0/RP0/CPU0:router(config-ospf)# segment-routing mpls	
Step 4	segment-routing forwarding mpls Example: RP/0/RP0/CPU0:router(config-ospf)# segment-routing forwarding mpls	Enables SR forwarding on all interfaces where this instance OSPF is enabled.
Step 5	segment-routing prefix-sid-map advertise-local Example: RP/0/RP0/CPU0:router(config-ospf)# segment-routing prefix-sid-map advertise local	Enables server functionality and allows OSPF to advertise the local mapping entries using area-scope flooding. The flooding is limited to areas where segment-routing is enabled. Disabled by default.
Step 6	segment-routing sr-prefer prefix-list[acl-name] Example: RP/0/RP0/CPU0:router(config-ospf)# segment-routing sr-prefer prefix-list foo	Enables OSPF to communicate to the routing information base (RIB) that SR labels are preferred to LDP labels. If ACL is used, OSPF signals the preference of SR labels over LDP labels for prefixes that match ACL. If ACL is not used, OSPF signals the preference of SR labels for all prefixes.

Example

The following example shows how OSPF advertises local mapping entries using area-flooding scope.

```

ipv4 prefix-list foo
permit 2.2.2.2/32
!
router ospf 1
router-id 1.1.1.1
segment-routing mpls
segment-routing forwarding mpls
segment-routing prefix-sid-map receive
segment-routing prefix-sid-map advertise-local
segment-routing sr-prefer prefix-list foo
area 0
interface Loopback0
prefix-sid index 1
!
interface GigabitEthernet0/0/0/0
!
interface GigabitEthernet0/2/0/0
!
interface GigabitEthernet0/2/0/3
!
!
area 1
interface GigabitEthernet0/2/0/7
!

```


Configuration Examples for Implementing OSPF

This section provides the following configuration examples:

Cisco IOS XR Software for OSPF Version 2 Configuration: Example

The following example shows how an OSPF interface is configured for an area in Cisco IOS XR Software.

area 0 must be explicitly configured with the **area** command and all interfaces that are in the range from 10.1.2.0 to 10.1.2.255 are bound to area 0. Interfaces are configured with the **interface** command (while the router is in area configuration mode) and the **area** keyword is not included in the interface statement.

Cisco IOS XR Software Configuration

```
interface GigabitEthernet 0/3/0/0
 ip address 10.1.2.1 255.255.255.255
 negotiation auto
!
router ospf 1
 router-id 10.2.3.4
 area 0
   interface GigabitEthernet 0/3/0/0
!
!
```

The following example shows how OSPF interface parameters are configured for an area in Cisco IOS XR software.

In Cisco IOS XR software, OSPF interface-specific parameters are configured in interface configuration mode and explicitly defined for area 0. In addition, the **ip ospf** keywords are no longer required.

Cisco IOS XR Software Configuration

```
interface GigabitEthernet 0/3/0/0
 ip address 10.1.2.1 255.255.255.0
 negotiation auto
!
router ospf 1
 router-id 10.2.3.4
 area 0
   interface GigabitEthernet 0/3/0/0
     cost 77
     mtu-ignore
     authentication message-digest
     message-digest-key 1 md5 0 test
!
!
```

The following example shows the hierarchical CLI structure of Cisco IOS XR software:

In Cisco IOS XR software, OSPF areas must be explicitly configured, and interfaces configured under the area configuration mode are explicitly bound to that area. In this example, interface 10.1.2.0/24 is bound to area 0 and interface 10.1.3.0/24 is bound to area 1.

Cisco IOS XR Software Configuration

```

interface GigabitEthernet 0/3/0/0
 ip address 10.1.2.1 255.255.255.0
 negotiation auto
!
interface GigabitEthernet 0/3/0/1
 ip address 10.1.3.1 255.255.255.0
 negotiation auto
!
router ospf 1
 router-id 10.2.3.4
 area 0
  interface GigabitEthernet 0/3/0/0
  !
 area 1
  interface GigabitEthernet 0/3/0/1
  !
 !
 !

```

CLI Inheritance and Precedence for OSPF Version 2: Example

The following example configures the cost parameter at different hierarchical levels of the OSPF topology, and illustrates how the parameter is inherited and how only one setting takes precedence. According to the precedence rule, the most explicit configuration is used.

The cost parameter is set to 5 in router configuration mode for the OSPF process. Area 1 sets the cost to 15 and area 6 sets the cost to 30. All interfaces in area 0 inherit a cost of 5 from the OSPF process because the cost was not set in area 0 or its interfaces.

In area 1, every interface has a cost of 15 because the cost is set in area 1 and 15 overrides the value 5 that was set in router configuration mode.

Area 4 does not set the cost, but GigabitEthernet interface 01/0/2 sets the cost to 20. The remaining interfaces in area 4 have a cost of 5 that is inherited from the OSPF process.

Area 6 sets the cost to 30, which is inherited by GigabitEthernet interfaces 0/1/0/3 and 0/2/0/3. GigabitEthernet interface 0/3/0/3 uses the cost of 1, which is set in interface configuration mode.

```

router ospf 1
 router-id 10.5.4.3
 cost 5
 area 0
  interface GigabitEthernet 0/1/0/0
  !
  interface GigabitEthernet 0/2/0/0
  !
  interface GigabitEthernet 0/3/0/0
  !
 !
 area 1
  cost 15
  interface GigabitEthernet 0/1/0/1
  !
  interface GigabitEthernet 0/2/0/1
  !
  interface GigabitEthernet 0/3/0/1
  !
 !
 area 4

```

```

interface GigabitEthernet 0/1/0/2
  cost 20
!
interface GigabitEthernet 0/2/0/2
!
interface GigabitEthernet 0/3/0/2
!
!
area 6
  cost 30
  interface GigabitEthernet 0/1/0/3
  !
  interface GigabitEthernet 0/2/0/3
  !
  interface GigabitEthernet 0/3/0/3
    cost 1
  !
!
!

```

MPLS TE for OSPF Version 2: Example

The following example shows how to configure the OSPF portion of MPLS TE. However, you still need to build an MPLS TE topology and create an MPLS TE tunnel. See the *MPLS Configuration Guide for the Cisco CRS Routers* for information.

In this example, loopback interface 0 is associated with area 0 and MPLS TE is configured within area 0.

```

interface Loopback 0
  address 10.10.10.10 255.255.255.0
!
interface GigabitEthernet 0/2/0/0
  address 10.1.2.2 255.255.255.0
!
router ospf 1
  router-id 10.10.10.10
  nsf
  auto-cost reference-bandwidth 10000
  mpls traffic-eng router-id Loopback 0
  area 0
    mpls traffic-eng
    interface GigabitEthernet 0/2/0/0
    interface Loopback 0

```

ABR with Summarization for OSPFv3: Example

The following example shows the prefix range 2300::/16 summarized from area 1 into the backbone:

```

router ospfv3 1
  router-id 192.168.0.217
  area 0
    interface GigabitEthernet 0/2/0/1
  area 1
    range 2300::/16
  interface GigabitEthernet 0/2/0/0

```

ABR Stub Area for OSPFv3: Example

The following example shows that area 1 is configured as a stub area:

```
router ospfv3 1
router-id 10.0.0.217
area 0
interface GigabitEthernet 0/2/0/1
area 1
stub
interface GigabitEthernet 0/2/0/0
```

ABR Totally Stub Area for OSPFv3: Example

The following example shows that area 1 is configured as a totally stub area:

```
router ospfv3 1
router-id 10.0.0.217
area 0
interface GigabitEthernet 0/2/0/1
area 1
stub no-summary
interface GigabitEthernet 0/2/0/0
```

Configuring OSPF SPF Prefix Prioritization: Example

This example shows how to configure /32 prefixes as medium-priority, in general, in addition to placing some /32 and /24 prefixes in critical-priority and high-priority queues:

```
prefix-set ospf-critical-prefixes
192.41.5.41/32,
11.1.3.0/24,
192.168.0.44/32
end-set
!
prefix-set ospf-high-prefixes
44.4.10.0/24,
192.41.4.41/32,
41.4.41.41/32
end-set
!
prefix-set ospf-medium-prefixes
0.0.0.0/0 ge 32
end-set
!

route-policy ospf-priority
if destination in ospf-high-prefixes then
set spf-priority high
else
if destination in ospf-critical-prefixes then
set spf-priority critical
else
if destination in ospf-medium-prefixes then
set spf-priority medium
endif
endif
```

```

endif
end-policy

```

OSPFv2

```

router ospf 1
  spf prefix-priority route-policy ospf-priority
  area 0
    interface GigabitEthernet0/3/0/0
    !
  area 3
    interface GigabitEthernet0/2/0/0
    !
  area 8
    interface GigabitEthernet0/2/0/0.590

```

OSPFv3

```

router ospfv3 1
  spf prefix-priority route-policy ospf-priority
  area 0
    interface GigabitEthernet0/3/0/0
    !
  area 3
    interface GigabitEthernet0/2/0/0
    !
  area 8
    interface GigabitEthernet0/2/0/0.590

```

Route Redistribution for OSPFv3: Example

The following example uses prefix lists to limit the routes redistributed from other protocols.

Only routes with 9898:1000 in the upper 32 bits and with prefix lengths from 32 to 64 are redistributed from BGP 42. Only routes *not* matching this pattern are redistributed from BGP 1956.

```

ipv6 prefix-list list1
  seq 10 permit 9898:1000::/32 ge 32 le 64
ipv6 prefix-list list2
  seq 10 deny 9898:1000::/32 ge 32 le 64
  seq 20 permit ::/0 le 128
router ospfv3 1
  router-id 10.0.0.217
  redistribute bgp 42
  redistribute bgp 1956
  distribute-list prefix-list list1 out bgp 42
  distribute-list prefix-list list2 out bgp 1956
  area 1
    interface GigabitEthernet 0/2/0/0

```

Virtual Link Configured Through Area 1 for OSPFv3: Example

This example shows how to set up a virtual link to connect the backbone through area 1 for the OSPFv3 topology that consists of areas 0 and 1 and virtual links 10.0.0.217 and 10.0.0.212:

ABR 1 Configuration

```
router ospfv3 1
router-id 10.0.0.217
area 0
 interface GigabitEthernet 0/2/0/1
area 1
 virtual-link 10.0.0.212
 interface GigabitEthernet 0/2/0/0
```

ABR 2 Configuration

```
router ospfv3 1
router-id 10.0.0.212
area 0
 interface GigabitEthernet 0/3/0/1
area 1
 virtual-link 10.0.0.217
 interface GigabitEthernet 0/2/0/0
```

Check the virtual links:

```
show ospfv3 virtual-links
Mon Dec 17 11:18:29.249 EST
```

```
Virtual Link OSPF_VL0 to router 192.168.0.4 is up
Interface ID 1000000, IPv6 address 13:13:13::4
Run as demand circuit
DoNotAge LSA allowed.
Transit area 1, via interface GigabitEthernet0/0/0/0, Cost of using 2
Transmit Delay is 1 sec, State POINT_TO_POINT,
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:06
Adjacency State INIT (Hello suppressed)
Index 0/0/0, retransmission queue length 0, number of retransmission 0
First 0(0)/0(0)/0(0) Next 0(0)/0(0)/0(0)
Last retransmission scan length is 0, maximum is 0
Last retransmission scan time is 0 msec, maximum is 0 msec
```

Virtual Link Configured with MD5 Authentication for OSPF Version 2: Example

The following examples show how to configure a virtual link to your backbone and apply MD5 authentication. You must perform the steps described on both ABRs at each end of the virtual link.

After you explicitly configure the ABRs, the configuration is inherited by all interfaces bound to that area—unless you override the values and configure them explicitly for the interface.

To understand virtual links, see [Virtual Link and Transit Area for OSPF, on page 368](#).

In this example, all interfaces on router ABR1 use MD5 authentication:

```
router ospf ABR1
router-id 10.10.10.10
```

```

authentication message-digest
message-digest-key 100 md5 0 cisco
area 0
 interface GigabitEthernet 0/2/0/1
 interface GigabitEthernet 0/3/0/0
area 1
 interface GigabitEthernet 0/3/0/1
virtual-link 10.10.5.5
!
!

```

In this example, only area 1 interfaces on router ABR3 use MD5 authentication:

```

router ospf ABR2
router-id 10.10.5.5
area 0
area 1
 authentication message-digest
message-digest-key 100 md5 0 cisco
 interface GigabitEthernet 0/9/0/1
virtual-link 10.10.10.10
area 3
 interface Loopback 0
 interface GigabitEthernet 0/9/0/0
!

```

VPN Backbone and Sham Link Configured for OSPF Version 2: Example

The following examples show how to configure a provider edge (PE) router to establish a VPN backbone and sham link connection:

```

logging console debugging
vrf vrf_1
 address-family ipv4 unicast
 import route-target
 100:1
 !
 export route-target
 100:1
 !
!
!
interface Loopback0
 ipv4 address 2.2.2.1 255.255.255.255
!
interface Loopback1
 vrf vrf_1
 ipv4 address 10.0.1.3 255.255.255.255
!
interface GigabitEthernet0/2/0/2
 vrf vrf_1
 ipv4 address 100.10.10.2 255.255.255.0
!
interface GigabitEthernet0/2/0/3
 ipv4 address 100.20.10.2 255.255.255.0
!
!
route-policy pass-all
 pass
end-policy
!

```

```

router ospf 1
log adjacency changes
router-id 2.2.2.2
vrf vrf_1
  router-id 22.22.22.2
  domain-id type 0005 value 111122223333
  domain-tag 140
  nsf ietf
  redistribute bgp 10
  area 0
    sham-link 10.0.1.3 10.0.0.101
    !
    interface GigabitEthernet0/2/0/2
    !
  !
!
!
router ospf 2
router-id 2.22.2.22
area 0
  interface Loopback0
  !
  interface GigabitEthernet0/2/0/3
  !
  !
!
router bgp 10
bgp router-id 2.2.2.1
bgp graceful-restart restart-time 300
bgp graceful-restart
address-family ipv4 unicast
  redistribute connected
  !
address-family vpnv4 unicast
  !
neighbor 2.2.2.2
remote-as 10
update-source Loopback0
address-family ipv4 unicast
  !
address-family vpnv4 unicast
  !
!
vrf vrf_1
  rd 100:1
  address-family ipv4 unicast
    redistribute connected route-policy pass-all
    redistribute ospf 1 match internal external
  !
  !
mpls ldp
  router-id 2.2.2.1
  interface GigabitEthernet0/2/0/3
  !
!

```

The show command for the configuration is as follows:

```

show ospf vrf all-inclusive sham-links
Mon Dec 17 10:27:41.815 EST

```

```

Sham Links for OSPF 1, VRF vrf1

```



```

Sham Link OSPF_SL0 to address 3.3.3.3 is up
Area 1, source address 1.1.1.1
IfIndex = 3
Run as demand circuit
DoNotAge LSA allowed., Cost of using 2
Transmit Delay is 1 sec, State POINT_TO_POINT,
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:08:911
Adjacency State FULL (Hello suppressed)
Number of DBD retrans during last exchange 0
Index 2/2, retransmission queue length 0, number of retransmission 1
First 0(0)/0(0) Next 0(0)/0(0)
Last retransmission scan length is 1, maximum is 1
Last retransmission scan time is 0 msec, maximum is 0 msec
Keychain-based authentication enabled
Keychain name key1
Key id used is 1
Cryptographic algorithm MD5_16

```

OSPF v3 Sham Links

The following example shows a configuration for ospfv3 sham links:

```

router ospfv3 1
vrf vrf1
auto-cost reference-bandwidth 1000
router-id 1.1.1.1
redistribute bgp 100 route-policy vrf1_rpl
area 1
sham-link 1111::1111 3333::3333
cost 2
!

```

The show command for the configuration is as follows:

```

show ospfv3 vrf all-inclusive sham-links
Mon Dec 17 11:06:05.192 EST

Sham Links for OSPFv3 1, VRF vrf1

Sham Link OSPF_SL0 to address 3333::3333 is up
Area 1, source address 1111::1111
IfIndex = 3
Run as demand circuit
DoNotAge LSA allowed., Cost of using 2
Transmit Delay is 1 sec, State POINT_TO_POINT,
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:08
Adjacency State FULL (Hello suppressed)
Number of DBD retrans during last exchange 0
Index 2/2, retransmission queue length 0, number of retransmission 1
First 0(0)/0(0) Next 0(0)/0(0)
Last retransmission scan length is 1, maximum is 1
Last retransmission scan time is 0 msec, maximum is 0 msec

```

OSPF Queue Tuning Parameters Configuration: Example

The following example shows how to configure the OSPF queue tuning parameters:

```

router ospf 100
queue dispatch incoming 30
queue limit high 1500

```

```
queue dispatch rate-limited-lsa 1000
queue dispatch spf-lsa-limit 2000
```

Where to Go Next

To configure route maps through the RPL for OSPF Version 2, see *Implementing Routing Policy on Cisco IOS XR Software* module.

To build an MPLS TE topology, create tunnels, and configure forwarding over the tunnel for OSPF Version 2; see *MPLS Configuration Guide for the Cisco CRS Routers*.

Additional References

The following sections provide references related to implementing OSPF.

Related Documents

Related Topic	Document Title
OSPF Commands and OSPFv3 Commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Routing Command Reference for Cisco CRS Routers</i>
MPLS TE feature information	<i>Implementing MPLS Traffic Engineering on Cisco IOS XR Software</i> module in <i>MPLS Configuration Guide for the Cisco CRS Routers</i>
MIB Reference	<i>Cisco Carrier Routing System and Cisco XR 12000 Series Router MIB Support Guide</i>

Standards

Standards	Title
draft-ietf-ospf-multi-area-adj-07.txt	OSPF Multi-Area Adjacency
draft-ietf-pce-disco-proto-ospf-08.txt	OSPF Protocol Extensions for Path Computation Element (PCE)
draft-ietf-mpls-igp-sync-00.txt	LDP IGP Synchronization
draft-ietf-ospf-ospfv3-graceful-restart-07.txt	OSPFv3 Graceful Restart

MIBs

MIBs	MIBs Link
—	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
RFC 1587	The OSPF NSSA Option
RFC 1793	Extending OSPF to Support Demand Circuits
RFC 2328	OSPF Version 2
RFC 2370	The OSPF Opaque LSA Option
RFC 2740	OSPF for IPv6
RFC 3101	The OSPF Not-So-Stubby Area (NSSA) Option
RFC 3137	OSPF Stub Router Advertisement
RFC 3509	Alternative Implementations of OSPF Area Border Routers
RFC 3623	Graceful OSPF Restart
RFC 3630	Traffic Engineering (TE) Extensions to OSPF Version 2
RFC 3682	The Generalized TTL Security Mechanism (GTSM)
RFC 3906	Calculating Interior Gateway Protocol (IGP) Routes Over Traffic Engineering Tunnels
RFC 4136	OSPF Refresh and Flooding Reduction in Stable Topologies
RFC 4206	Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)

RFCs	Title
RFC 4124	Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering
RFC 4576	Using a Link State Advertisement (LSA) Options Bit to Prevent Looping in BGP/MPLS IP Virtual Private Networks (VPNs) ownbit Extension for L3VPN
RFC 4577	OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)
RFC 4750	OSPF Version 2 Management Information Base
RFC 4811	OSPF Out-of-Band Link State Database (LSDB) Resynchronization
RFC 4812	OSPF Restart Signaling
RFC 4813	OSPF Link-Local Signaling
RFC 4970	Extensions to OSPF for Advertising Optional Router Capabilities
RFC 5643	Management Information Base (MIB) for OSPFv3

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport



CHAPTER 8

Implementing IP Fast Reroute Loop-Free Alternate

IP Fast Reroute Loop-Free Alternate feature enables you to tunnel a packet around a failed link to a remote loop-free alternate that is more than one hop away.

- [Prerequisites for IPv4/IPv6 Loop-Free Alternate Fast Reroute, on page 447](#)
- [Restrictions for Loop-Free Alternate Fast Reroute, on page 447](#)
- [IS-IS and IP FRR, on page 448](#)
- [Repair Paths, on page 448](#)
- [LFA Overview, on page 449](#)
- [LFA Calculation, on page 449](#)
- [Interaction Between RIB and Routing Protocols, on page 449](#)
- [Configuring Fast Reroute Support, on page 450](#)
- [Configuring IPv4 Loop-Free Alternate Fast Reroute Support: Example, on page 452](#)
- [Additional References, on page 452](#)

Prerequisites for IPv4/IPv6 Loop-Free Alternate Fast Reroute

- Loop-Free Alternate (LFA) Fast Reroute (FRR) can protect paths that are reachable through an interface only if the interface is a point-to-point interface.
- When a LAN interface is physically connected to a single neighbor, you should configure the LAN interface as a point-to-point interface so that it can be protected through LFA FRR.

Restrictions for Loop-Free Alternate Fast Reroute

- Load balance support is available for FRR-protected prefixes, but the 50 ms cutover time is not guaranteed.
- A maximum of eight FRR-protected interfaces can simultaneously undergo a cutover.
- Only Layer 3 VPN is supported.
- The remote LFA backup path for MPLS traffic can be setup only using LDP.

- LFA calculations are restricted to interfaces or links belonging to the same level or area. Hence, excluding all neighbors on the same LAN when computing the backup LFA can result in repairs being unavailable in a subset of topologies.
- Only physical and physical port-channel interfaces are protected. Subinterfaces, tunnels, and virtual interfaces are not protected.
- Border Gateway Protocol (BGP) Prefix-Independent Convergence (PIC) and IP FRR can be configured on the same interface as long as they are not used for the same prefix.
- IPv6 LFA FRR not supported over TE tunnel.

IS-IS and IP FRR

When a local link fails in a network, IS-IS recomputes new primary next-hop routes for all affected prefixes. These prefixes are updated in the RIB and the Forwarding Information Base (FIB). Until the primary prefixes are updated in the forwarding plane, traffic directed towards the affected prefixes are discarded. This process can take hundreds of milliseconds.

In IP FRR, IS-IS computes LFA next-hop routes for the forwarding plane to use in case of primary path failures. LFA is computed per prefix.

When there are multiple LFAs for a given primary path, IS-IS uses a tiebreaking rule to pick a single LFA for a primary path. In case of a primary path with multiple LFA paths, prefixes are distributed equally among LFA paths.

Repair Paths

Repair paths forward traffic during a routing transition. When a link or a router fails, due to the loss of a physical layer signal, initially, only the neighboring routers are aware of the failure. All other routers in the network are unaware of the nature and location of this failure until information about this failure is propagated through a routing protocol, which may take several hundred milliseconds. It is, therefore, necessary to arrange for packets affected by the network failure to be steered to their destinations.

A router adjacent to the failed link employs a set of repair paths for packets that would have used the failed link. These repair paths are used from the time the router detects the failure until the routing transition is complete. By the time the routing transition is complete, all routers in the network revise their forwarding data and the failed link is eliminated from the routing computation.

Repair paths are precomputed in anticipation of failures so that they can be activated the moment a failure is detected.

The LFA FRR feature uses the following repair paths:

- Equal Cost Multipath (ECMP) uses a link as a member of an equal cost path-split set for a destination. The other members of the set can provide an alternative path when the link fails.
- LFA is a next-hop route that delivers a packet to its destination without looping back. Downstream paths are a subset of LFAs.

LFA Overview

LFA is a node other than the primary neighbor. Traffic is redirected to an LFA after a network failure. An LFA makes the forwarding decision without any knowledge of the failure.

An LFA must neither use a failed element nor use a protecting node to forward traffic. An LFA must not cause loops. By default, LFA is enabled on all supported interfaces as long as the interface can be used as a primary path.

Advantages of using per-prefix LFAs are as follows:

- The repair path forwards traffic during transition when the primary path link is down.
- All destinations having a per-prefix LFA are protected. This leaves only a subset (a node at the far side of the failure) unprotected.

LFA Calculation

The general algorithms to compute per-prefix LFAs can be found in RFC 5286. IS-IS implements RFC 5286 with a small change to reduce memory usage. Instead of performing a Shortest Path First (SPF) calculation for all neighbors before examining prefixes for protection, IS-IS examines prefixes after SPF calculation is performed for each neighbor. Because IS-IS examines prefixes after SPF calculation is performed, IS-IS retains the best repair path after SPF calculation is performed for each neighbor. IS-IS does not have to save SPF results for all neighbors.

Interaction Between RIB and Routing Protocols

A routing protocol computes repair paths for prefixes by implementing tiebreaking algorithms. The end result of the computation is a set of prefixes with primary paths, where some primary paths are associated with repair paths.

A tiebreaking algorithm considers LFAs that satisfy certain conditions or have certain attributes. When there is more than one LFA, configure the **fast-reroute per-prefix** command with the **tie-break** keyword. If a rule eliminates all candidate LFAs, then the rule is skipped.

A primary path can have multiple LFAs. A routing protocol is required to implement default tiebreaking rules and to allow you to modify these rules. The objective of the tiebreaking algorithm is to eliminate multiple candidate LFAs, select one LFA per primary path per prefix, and distribute the traffic over multiple candidate LFAs when the primary path fails.

Tiebreaking rules cannot eliminate all candidates.

The following attributes are used for tiebreaking:

- Downstream—Eliminates candidates whose metric to the protected destination is lower than the metric of the protecting node to the destination.
- Linecard-disjoint—Eliminates candidates sharing the same linecard with the protected path.
- Shared Risk Link Group (SRLG)—Eliminates candidates that belong to one of the protected path SRLGs.
- Load-sharing—Distributes remaining candidates among prefixes sharing the protected path.

- Lowest-repair-path-metric—Eliminates candidates whose metric to the protected prefix is higher.
- Node protecting—Eliminates candidates that are not node protected.
- Primary-path—Eliminates candidates that are not ECMPs.
- Secondary-path—Eliminates candidates that are ECMPs.

Configuring Fast Reroute Support



Note LFA computations are enabled for all routes and FRR is enabled for all supported interfaces.

SUMMARY STEPS

1. **configure**
2. **router isis** *process-id*
3. **is-type** { level-1 | level-1-2 | level-2-only }
4. **net** *net*
5. **address-family** { ipv4 | ipv6 } [unicast | multicast]
6. **metric-style** wide
7. **exit**
8. **interface** *bundle bundle-id*
9. **address-family** { ipv4 | ipv6 } [unicast | multicast]
10. **fast-reroute per-prefix**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure Example: RP/0/RP0/CPU0:router# configure	Enters global configuration mode.
Step 2	router isis <i>process-id</i> Example: RP/0/RP0/CPU0:router(config-if)# router isis core	Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode. By default, all IS-IS instances are automatically at Level 1 and Level 2. You can change this level by a particular routing instance using the is-type router configuration command.
Step 3	is-type { level-1 level-1-2 level-2-only } Example: RP/0/RP0/CPU0:router(config-isis)#is-type level-2-only	(Optional) Configures the system type (area or backbone router). By default, every IS-IS instance acts as a level-1-2 router. <ul style="list-style-type: none"> • The level-1 keyword configures the software to perform only Level 1 (intra-area) routing. Only Level 1 adjacencies are established. The software only

	Command or Action	Purpose
		<p>detects destinations within its area. Packets containing destinations outside the area if any, are sent to the nearest level-1-2 router in the area.</p> <ul style="list-style-type: none"> • The level-2-only keyword configures the software to perform Level 2 (backbone) routing only, the router only establishes Level 2 adjacencies. It is established either with other Level 2-only routers or with level-1-2 routers. • The level-1-2 keyword configures the software to perform both Level 1 and Level 2 routing. Both Level 1 and Level 2 adjacencies are established. The router acts as a border router between the Level 2 backbone and its Level 1 area.
Step 4	<p>net <i>net</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-isis)# net 47.0001.0000.0000.8888.00</pre>	Configures an IS-IS network entity (NET) for a routing process.
Step 5	<p>address-family {<i>ipv4</i> <i>ipv6</i>} [unicast multicast]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast</pre>	Specifies the IPv4 or IPv6 address family, and enters the interface address family configuration mode.
Step 6	<p>metric-style <i>wide</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide</pre>	Configures a router to generate and accept wide link metrics only.
Step 7	<p>exit</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-isis-af)# exit</pre>	Exits router address family configuration mode, and resets the router to router configuration mode.
Step 8	<p>interface <i>bundle bundle-id</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-isis)# interface Bundle-Ether 9</pre>	Creates and names a new Ethernet link bundle.
Step 9	<p>address-family {<i>ipv4</i> <i>ipv6</i>} [unicast multicast]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast</pre>	Specifies the IPv4 or IPv6 address family, and enters interface address family configuration mode.
Step 10	<p>fast-reroute per-prefix</p> <p>Example:</p>	Enables per-prefix FRR.

Command or Action	Purpose
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix	

Configuring IPv4 Loop-Free Alternate Fast Reroute Support: Example

The following example shows how to configure IPv4 LFA FRR.

```
router isis core
 is-type level-2-only
 net 47.0001.0000.0000.8888.00
 address-family ipv4 unicast
  metric-style wide
 exit
 !
 interface Bundle-Ether 9
  point-to-point
  address-family ipv4 unicast
   fast-reroute per-prefix
 !
 !
```

Additional References

The following sections provide references related to implementing IPv4/IPv6 Loop-Free Alternate Fast Reroute.

Related Documents

Related Topic	Document Title
IS-IS commands	<i>Routing Command Reference for Cisco CRS Routers</i>
MPLS commands	<i>Routing Command Reference for Cisco CRS Routers</i>

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport



CHAPTER 9

Implementing and Monitoring RIB

Routing Information Base (RIB) is a distributed collection of information about routing connectivity among all nodes of a network. Each router maintains a RIB containing the routing information for that router. RIB stores the best routes from all routing protocols that are running on the system.

This module describes how to implement and monitor RIB on Cisco IOS XR network.



Note For more information about RIB on the Cisco IOS XR software and complete descriptions of RIB commands listed in this module, see the Additional References section of this module.

To locate documentation for other commands that might appear during the execution of a configuration task, search online in the *Cisco IOS XR Commands Master List for the Cisco CRS Router*.

Feature History for Implementing and Monitoring RIB

Release	Modification
Release 2.0	This feature was introduced.
Release 3.3.0	VPN routing and forwarding (VRF) support was added to the command syntax.
Release 3.4.0	RIB statistics support was added using the show rib statistics command. Disabling RIB next-hop dampening was supported.
Release 3.5.0	The following features were supported: <ul style="list-style-type: none">• IP fast reroute loop-free alternates computation• IPv6 Provider Edge and IPv6 VPN Provider Edge over Multiprotocol Label Switching• RIB quarantining
Release 4.2.0	The following features were added: <ul style="list-style-type: none">• Route and Label Consistency Checker (RCC and LCC)
Release 4.2.1	BGP Prefix Independent Convergence for RIB and FIB support was added.

- [Prerequisites for Implementing RIB, on page 456](#)

- [Information About RIB Configuration, on page 456](#)
- [How to Deploy and Monitor RIB, on page 460](#)
- [Configuring RCC and LCC, on page 463](#)
- [Configuration Examples for RIB Monitoring, on page 465](#)
- [Where to Go Next, on page 468](#)
- [Additional References, on page 469](#)

Prerequisites for Implementing RIB

- 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.
- RIB is distributed with the base Cisco IOS XR software; as such, it does not have any special requirements for installation. The following are the requirements for base software installation:
 - Router
 - Cisco IOS XR software
 - Base package

Information About RIB Configuration

To implement the Cisco RIB feature, you must understand the following concepts:

Overview of RIB

Each routing protocol selects its own set of best routes and installs those routes and their attributes in RIB. RIB stores these routes and selects the best ones from among all routing protocols. Those routes are downloaded to the line cards for use in forwarding packets. The acronym RIB is used both to refer to RIB processes and the collection of route data contained within RIB.

Within a protocol, routes are selected based on the metrics in use by that protocol. A protocol downloads its best routes (lowest or tied metric) to RIB. RIB selects the best overall route by comparing the administrative distance of the associated protocol.

RIB Data Structures in BGP and Other Protocols

RIB uses processes and maintains data structures distinct from other routing applications, such as Border Gateway Protocol (BGP) and other unicast routing protocols, or multicast protocols, such as Protocol Independent Multicast (PIM) or Multicast Source Discovery Protocol (MSDP). However, these routing protocols use internal data structures similar to what RIB uses, and may internally refer to the data structures as a RIB. For example, BGP routes are stored in the BGP RIB (BRIB), and multicast routes, computed by multicast routing protocols such as PIM and MSDP, are stored in the Multicast RIB (MRIB). RIB processes are not responsible for the BRIB and MRIB, which are handled by BGP and multicast processes, respectively.

The table used by the line cards and RP to forward packets is called the Forwarding Information Base (FIB). RIB processes do not build the FIBs. Instead, RIB downloads the set of selected best routes to the FIB processes, by the Bulk Content Downloader (BCDL) process, onto each line card. FIBs are then constructed.

RIB Administrative Distance

Forwarding is done based on the longest prefix match. If you are forwarding a packet destined to 10.0.2.1, you prefer 10.0.2.0/24 over 10.0.0.0/16 because the mask /24 is longer (and more specific) than a /16.

Routes from different protocols that have the same prefix and length are chosen based on administrative distance. For instance, the Open Shortest Path First (OSPF) protocol has an administrative distance of 110, and the Intermediate System-to-Intermediate System (IS-IS) protocol has an administrative distance of 115. If IS-IS and OSPF both download 10.0.1.0/24 to RIB, RIB would prefer the OSPF route because OSPF has a lower administrative distance. Administrative distance is used only to choose between multiple routes of the same length.

This table lists default administrative distances for the common protocols.

Table 6: Default Administrative Distances

Protocol	Administrative Distance Default
Connected or local routes	0
Static routes	1
External BGP routes	20
OSPF routes	110
IS-IS routes	115
Internal BGP routes	200

The administrative distance for some routing protocols (for instance IS-IS, OSPF, and BGP) can be changed. See the protocol-specific documentation for the proper method to change the administrative distance of that protocol.



Note Changing the administrative distance of a protocol on some but not all routers can lead to routing loops and other undesirable behavior. Doing so is not recommended.

RIB Support for IPv4 and IPv6

In Cisco IOS XR software, RIB tables support multicast and unicast routing.

The default routing tables for Cisco IOS XR software RIB are the unicast RIB tables for IPv4 routing and the multicast-unicast RIB tables for IPv6 routing. For multicast routing, routing protocols insert unicast routes into the multicast-unicast RIB table. Multicast protocols then use the information to build multicast routes (which in turn are stored in the MRIB). See the multicast documentation for more information on using and configuring multicast.

RIB processes `ipv4_rib` and `ipv6_rib` run on the RP card. If process placement functionality is available and supported by multiple RPs in the router, RIB processes can be placed on any available node.

RIB Statistics

RIB supports statistics for messages (requests) flowing between the RIB and its clients. Protocol clients send messages to the RIB (for example, route add, route delete, and next-hop register, and so on). RIB also sends messages (for example, redistribute routes, advertisements, next-hop notifications, and so on). These statistics are used to gather information about what messages have been sent and the number of messages that have been sent. These statistics provide counters for the various messages that flow between the RIB server and its clients. The statistics are displayed using the **show rib statistics** command.

RIB maintains counters for all requests sent from a client including:

- Route operations
- Table registrations
- Next-hop registrations
- Redistribution registrations
- Attribute registrations
- Synchronization completion

RIB also maintains counters for all requests sent by the RIB. The configuration will disable the RIB next-hop dampening feature. As a result, RIB notifies client immediately when a next hop that client registered for is resolved or unresolved.

RIB also maintains the results of the requests.

IPv6 Provider Edge IPv6 and IPv6 VPN Provider Edge Transport over MPLS

IPv6 Provider Edge (6PE) and IPv6 VPN Provider Edge (6VPE) leverages the existing Multiprotocol Label Switching (MPLS) IPv4 core infrastructure for IPv6 transport. 6PE and 6VPE enables IPv6 sites to communicate with each other over an MPLS IPv4 core network using MPLS label switched paths (LSPs).

RIB supports 6PE and 6VPE by providing 6VPE next hops. The next-hop information is stored in an opaque database in RIB, which is populated by protocol clients with data to be sent to the Forwarding Information Base (FIB).

For detailed information about configuring 6PE and 6VPE over MPLS, see *MPLS Configuration Guide for the Cisco CRS Routers*.

IP Fast Reroute

The IP Fast Reroute (IPFRR) loop-free alternate (LFA) computation provides protection against link failure. Locally computed repair paths are used to prevent packet loss caused by loops that occur during network reconvergence after a failure. For information about IPFRR see *Implementing IS-IS on Cisco IOS XR Software* module in *Routing Configuration Guide for Cisco CRS Routers*.

RIB Quarantining

RIB quarantining solves the problem in the interaction between routing protocols and the RIB. The problem is a persistent oscillation between the RIB and routing protocols that occurs when a route is continuously inserted and then withdrawn from the RIB, resulting in a spike in CPU use until the problem is resolved. If there is no damping on the oscillation, then both the protocol process and the RIB process have high CPU use, affecting the rest of the system as well as blocking out other protocol and RIB operations. This problem occurs when a particular combination of routes is received and installed in the RIB. This problem typically happens as a result of a network misconfiguration. However, because the misconfiguration is across the network, it is not possible to detect the problem at configuration time on any single router.

The quarantining mechanism detects mutually recursive routes and quarantines the last route that completes the mutual recursion. The quarantined route is periodically evaluated to see if the mutual recursion has gone away. If the recursion still exists, the route remains quarantined. If the recursion has gone away, the route is released from its quarantine.

The following steps are used to quarantine a route:

1. RIB detects when a particular problematic path is installed.
2. RIB sends a notification to the protocol that installed the path.
3. When the protocol receives the quarantine notification about the problem route, it marks the route as being “quarantined.” If it is a BGP route, BGP does not advertise reachability for the route to its neighbors.
4. Periodically, RIB tests all its quarantined paths to see if they can now safely be installed (moved from quarantined to "Ok to use" state). A notification is sent to the protocol to indicate that the path is now safe to use.

Route and Label Consistency Checker

The Route Consistency Checker and Label Consistency Checker (RCC/LCC) are command-line tools that can be used to verify consistency between control plane and data plane route and label programming in IOS XR software.

Routers in production networks may end up in a state where the forwarding information does not match the control plane information. Possible causes of this include fabric or transport failures between the Route Processor (RP) and the line cards (LCs), or issues with the Forwarding Information Base (FIB). RCC/LCC can be used to identify and provide detailed information about resultant inconsistencies between the control plane and data plane. This information can be used to further investigate and diagnose the cause of forwarding problems and traffic loss.

RCC/LCC can be run in two modes. It can be triggered from EXEC mode as an on-demand, one-time scan (On-demand Scan), or be configured to run at defined intervals in the background during normal router operation (Background Scan). RCC compares the Routing Information Base (RIB) against the Forwarding Information Base (FIB) while LCC compares the Label Switching Database (LSD) against the FIB. When an inconsistency is detected, RCC/LCC output will identify the specific route or label and identify the type of inconsistency detected as well as provide additional data that will assist with further troubleshooting.

RCC runs on the Route Processor. FIB checks for errors on the line card and forwards first the 20 error reports to RCC. RCC receives error reports from all nodes, summarizes them (checks for exact match), and adds it to two queues, soft or hard. Each queue has a limit of 1000 error reports and there is no prioritization in the queue. RCC/LCC logs the same errors (exact match) from different nodes as one error. RCC/LCC compares the errors based on prefix/label, version number, type of error, etc.

On-demand Scan

In On-demand Scan, user requests scan through the command line interface on a particular prefix in a particular table or all the prefixes in the table. The scan is run immediately and the results are published right away. LCC performs on-demand scan on the LSD, where as RCC performs it per VRF.

Background Scan

In Background Scan, user configures the scan that is then left to run in the background. The configuration consists of the time period for the periodic scan. This scan can be configured on either a single table or multiple tables. LCC performs background scan on the LSD, where as RCC performs it either for default or other VRFs.

Flex-LSR Label Switch Processor 140

The Label Switch Processor 140 line card is a scaled down version of Cisco CRS 3 line card. The Flex -LSR feature imposes these limits to download of routes and labels:

- 16K IPv4 and IPv6 unicast (including VRF) routes
- 8K IPv4 and IPv6 multicast routes
- 100K labels

How to Deploy and Monitor RIB

To deploy and monitor RIB, you must understand the following concepts:

Verifying RIB Configuration Using the Routing Table

Perform this task to verify the RIB configuration to ensure that RIB is running on the RP and functioning properly by checking the routing table summary and details.

SUMMARY STEPS

1. **show route** [vrf { *vrf-name* | all }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **summary** [**detail**] [**standby**]
2. **show route** [vrf { *vrf-name* | all }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] [*protocol* [*instance*] | *ip-address mask*] [**standby**] [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	<pre>show route [vrf { <i>vrf-name</i> all }] [afi-all ipv4 ipv6] [unicast multicast safi-all] summary [detail] [standby]</pre> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show route summary</pre>	<p>Displays route summary information about the specified routing table.</p> <ul style="list-style-type: none"> • The default table summarized is the IPv4 unicast routing table.

	Command or Action	Purpose
Step 2	<p>show route [vrf { vrf-name all }] [afi-all ipv4 ipv6] [unicast multicast safi-all] [protocol [instance] ip-address mask] [standby] [detail]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show route ipv4 unicast</pre>	<p>Displays more detailed route information about the specified routing table.</p> <ul style="list-style-type: none"> This command is usually issued with an IP address or other optional filters to limit its display. Otherwise, it displays all routes from the default IPv4 unicast routing table, which can result in an extensive list, depending on the configuration of the network.

Verifying Networking and Routing Problems

Perform this task to verify the operation of routes between nodes.

SUMMARY STEPS

- show route** [vrf { vrf-name | all }] [afi-all | ipv4 | ipv6] [unicast | multicast | safi-all] [protocol [instance] | ip-address mask] [standby] [detail]
- show route** [vrf { vrf-name | all }] [afi-all | ipv4 | ipv6] [unicast | multicast | safi-all] backup [ip-address] [standby]
- show route** [vrf { vrf-name | all }] [ipv4 | ipv6] [unicast | multicast | safi-all] best-local ip-address [standby]
- show route** [vrf { vrf-name | all }] [afi-all | ipv4 | ipv6] [unicast | multicast | safi-all] connected [standby]
- show route** [vrf { vrf-name | all }] [afi-all | ipv4 | ipv6] [unicast | multicast | safi-all] local [interface] [standby]
- show route** [vrf { vrf-name | all }] [ipv4 | ipv6] [unicast | multicast | safi-all] longer-prefixes { ip-address mask | ip-address / prefix-length } [standby]
- show route** [vrf { vrf-name | all }] [ipv4 | ipv6] [unicast | multicast | safi-all] next-hop ip-address [standby]

DETAILED STEPS

	Command or Action	Purpose
Step 1	<p>show route [vrf { vrf-name all }] [afi-all ipv4 ipv6] [unicast multicast safi-all] [protocol [instance] ip-address mask] [standby] [detail]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show route list list1 bgp aspo ipv4 unicast 192.168.1.11/8</pre>	Displays the current routes in RIB.
Step 2	<p>show route [vrf { vrf-name all }] [afi-all ipv4 ipv6] [unicast multicast safi-all] backup [ip-address] [standby]</p> <p>Example:</p>	Displays backup routes in RIB.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router# show route ipv4 unicast backup 192.168.1.11/8	
Step 3	<p>show route [vrf { <i>vrf-name</i> all }] [ipv4 ipv6] [unicast multicast safi-all] best-local <i>ip-address</i> [standby]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show route ipv4 unicast best-local 192.168.1.11/8</pre>	Displays the best-local address to use for return packets from the given destination.
Step 4	<p>show route [vrf { <i>vrf-name</i> all }] [afi-all ipv4 ipv6] [unicast multicast safi-all] connected [standby]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show route ipv4 unicast connected</pre>	Displays the current connected routes of the routing table.
Step 5	<p>show route [vrf { <i>vrf-name</i> all }] [afi-all ipv4 ipv6] [unicast multicast safi-all] local [<i>interface</i>] [standby]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show route ipv4 unicast local</pre>	Displays local routes for receive entries in the routing table.
Step 6	<p>show route [vrf { <i>vrf-name</i> all }] [ipv4 ipv6] [unicast multicast safi-all] longer-prefixes { <i>ip-address mask</i> <i>ip-address / prefix-length</i> } [standby]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show route ipv4 unicast longer-prefixes 192.168.1.11/8</pre>	Displays the current routes in RIB that share a given number of bits with a given network.
Step 7	<p>show route [vrf { <i>vrf-name</i> all }] [ipv4 ipv6] [unicast multicast safi-all] next-hop <i>ip-address</i> [standby]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# show route ipv4 unicast next-hop 192.168.1.34</pre>	Displays the next-hop gateway or host to a destination address.

Disabling RIB Next-hop Dampening

Perform this task to disable RIB next-hop dampening.

SUMMARY STEPS

1. **router rib**
2. **address-family { ipv4 | ipv6 } next-hop dampening disable**
3. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	router rib Example: RP/0/RP0/CPU0:router# route rib	Enters RIB configuration mode.
Step 2	address-family { ipv4 ipv6 } next-hop dampening disable Example: RP/0/RP0/CPU0:router(config-rib)# address family ipv4 next-hop dampening disable	Disables next-hop dampening for IPv4 address families.
Step 3	commit	

Configuring RCC and LCC

Enabling RCC and LCC On-demand Scan

Perform this task to trigger route consistency checker (RCC) and Label Consistency Checker (LCC) on-demand scan. The on-demand scan can be run on a particular address family (AFI), sub address family (SAFI), table and prefix, vrf, or all prefixes in the table.

SUMMARY STEPS

1. Use one of these commands.
 - **show rcc { ipv4 | ipv6 } unicast [all] [prefix/mask] [vrf vrf-name]**
 - **show lcc { ipv4 | ipv6 } unicast [all] [prefix/mask] [vrf vrf-name]**
2. Use one of these commands.
 - **clear rcc { ipv4 | ipv6 } unicast [all] [prefix/mask] [vrf vrf-name] log**
 - **clear lcc { ipv4 | ipv6 } unicast [all] [prefix/mask] [vrf vrf-name] log**

DETAILED STEPS

	Command or Action	Purpose
Step 1	<p>Use one of these commands.</p> <ul style="list-style-type: none"> • show rcc {ipv4 ipv6} unicast [all] [prefix/mask] [vrf vrf-name] • show lcc {ipv4 ipv6} unicast [all] [prefix/mask] [vrf vrf-name] <p>Example:</p> <pre>RP/0/RP0/CPU0:router#show rcc ipv6 unicast 2001:DB8::/32 vrf vrf_1</pre> <p>Or</p> <pre>RP/0/RP0/CPU0:router#show lcc ipv6 unicast 2001:DB8::/32 vrf vrf_1</pre>	Runs on-demand Route Consistency Checker (RCC) or Label Consistency Checker (LCC).
Step 2	<p>Use one of these commands.</p> <ul style="list-style-type: none"> • clear rcc {ipv4 ipv6} unicast [all] [prefix/mask] [vrf vrf-name] log • clear lcc {ipv4 ipv6} unicast [all] [prefix/mask] [vrf vrf-name] log <p>Example:</p> <pre>RP/0/RP0/CPU0:router#clear rcc ipv6 unicast log</pre> <p>Or</p> <pre>RP/0/RP0/CPU0:router#show lcc ipv6 unicast log</pre>	Clears the log of previous scans.

Enabling RCC and LCC Background Scan

Perform this task to run a background scan for Route Consistency Checker (RCC) and Label Consistency Checker (LCC).

SUMMARY STEPS

- configure**
- Use one of these commands:
 - **rcc** {ipv4 | ipv6} unicast {enable | period milliseconds}
 - **lcc** {ipv4 | ipv6} unicast {enable | period milliseconds}
- commit**
- Use one of these commands:
 - **show rcc** {ipv4 | ipv6} unicast [summary | scan-id scan-id-value]
 - **show lcc** {ipv4 | ipv6} unicast [summary | scan-id scan-id-value]

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	<p>Use one of these commands:</p> <ul style="list-style-type: none"> • rcc {ipv4 ipv6} unicast {enable period milliseconds} • lcc {ipv4 ipv6} unicast {enable period milliseconds} <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config)#rcc ipv6 unicast enable</pre> <pre>RP/0/RP0/CPU0:router(config)#rcc ipv6 unicast period 500</pre> <p>Or</p> <pre>RP/0/RP0/CPU0:router(config)#lcc ipv6 unicast enable</pre> <pre>RP/0/RP0/CPU0:router(config)#lcc ipv6 unicast period 500</pre>	<p>Triggers RCC or LCC background scan. Use the period option to control how often the verification be triggered. Each time the scan is triggered, verification is resumed from where it was left out and one buffer's worth of routes or labels are sent to the forwarding information base (FIB).</p>
Step 3	commit	
Step 4	<p>Use one of these commands.</p> <ul style="list-style-type: none"> • show rcc {ipv4 ipv6} unicast [summary scan-id scan-id-value] • show lcc {ipv4 ipv6} unicast [summary scan-id scan-id-value] <p>Example:</p> <pre>RP/0/RP0/CPU0:router#show rcc ipv6 unicast statistics scan-id 120</pre> <p>Or</p> <pre>RP/0/RP0/CPU0:router#show lcc ipv6 unicast statistics scan-id 120</pre>	<p>Displays statistics about background scans.</p> <ul style="list-style-type: none"> • summary—Displays the current ongoing scan id and a summary of the previous few scans. • scan-id scan-id-value—Displays details about a specific scan.

Configuration Examples for RIB Monitoring

RIB is not configured separately for the Cisco IOS XR system. RIB computes connectivity of the router with other nodes in the network based on input from the routing protocols. RIB may be used to monitor and troubleshoot the connections between RIB and its clients, but it is essentially used to monitor routing connectivity between the nodes in a network. This section contains displays from the **show** commands used to monitor that activity.

Output of show route Command: Example

The following is sample output from the **show route** command when entered without an address:

```
show route
```

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local
```

```
Gateway of last resort is 172.23.54.1 to network 0.0.0.0
```

```
C   10.2.210.0/24 is directly connected, 1d21h, Ethernet0/1/0/0
L   10.2.210.221/32 is directly connected, 1d21h, Ethernet0/1/1/0
C   172.20.16.0/24 is directly connected, 1d21h, ATM4/0.1
L   172.20.16.1/32 is directly connected, 1d21h, ATM4/0.1
C   10.6.100.0/24 is directly connected, 1d21h, Loopback1
L   10.6.200.21/32 is directly connected, 1d21h, Loopback0
S   192.168.40.0/24 [1/0] via 172.20.16.6, 1d21h
```

Output of show route backup Command: Example

The following is sample output from the **show route backup** command:

```
show route backup
```

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local
S   172.73.51.0/24 is directly connected, 2d20h, GigabitEthernet 4/0/0/1
      Backup O E2 [110/1] via 10.12.12.2, GigabitEthernet 3/0/0/1
```

Output of show route best-local Command: Example

The following is sample output from the **show route best-local** command:

```
show route best-local 10.12.12.1
```

```
Routing entry for 10.12.12.1/32
  Known via "local", distance 0, metric 0 (connected)
  Routing Descriptor Blocks
    10.12.12.1 directly connected, via GigabitEthernet3/0
    Route metric is 0
```


Output of show route connected Command: Example

The following is sample output from the **show route connected** command:

```
show route connected

C    10.2.210.0/24 is directly connected, 1d21h, Ethernet0
C    172.20.16.0/24 is directly connected, 1d21h, ATM4/0.1
C    10.6.100.0/24 is directly connected, 1d21h, Loopback1
```

Output of show route local Command: Example

The following is sample output from the **show route local** command:

```
show route local

L    10.10.10.1/32 is directly connected, 00:14:36, Loopback0
L    10.91.36.98/32 is directly connected, 00:14:32, Ethernet0/0
L    172.22.12.1/32 is directly connected, 00:13:35, GigabitEthernet3/0
L    192.168.20.2/32 is directly connected, 00:13:27, GigabitEthernet2/0
L    10.254.254.1/32 is directly connected, 00:13:26, GigabitEthernet2/2
```

Output of show route longer-prefixes Command: Example

The following is sample output from the **show route longer-prefixes** command:

```
show route ipv4 longer-prefixes 172.16.0.0/8
longer-prefixes

Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
O - OSPF, IA - OSPF inter area, N1 - OSPF NSSA external type 1
N2 - OSPF NSSA external type 2, E1 - OSPF external type 1
E2 - OSPF external type 2, E - EGP, i - ISIS, L1 - IS-IS level-1
L2 - IS-IS level-2, ia - IS-IS inter area
su - IS-IS summary null, * - candidate default
U - per-user static route, o - ODR, L - local

Gateway of last resort is 172.23.54.1 to network 0.0.0.0
S    172.16.2.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.3.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.4.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.5.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.6.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.7.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.8.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.9.0/32 is directly connected, 00:00:24, Loopback0
```

Output of show route next-hop Command: Example

The following is sample output from the **show route resolving-next-hop** command:

```
show route resolving-next-hop 10.0.0.1
```

```

Nexthop matches 0.0.0.0/0
  Known via "static", distance 200, metric 0, candidate default path
  Installed Aug 18 00:59:04.448
  Directly connected nexthops
    172.29.52.1, via MgmtEth0/
RP0
 /CPU0/0
   Route metric is 0
 172.29.52.1, via MgmtEth0/RP1/CPU0/0
   Route metric is 0

```

Enabling RCC and LCC: Example

Enabling RCC and LCC Background Scan: Example

This example shows how to enable Route Consistency Checker (RCC) background scan with a period of 500 milliseconds between buffers in scans for IPv6 unicast tables:

```
rcc ipv6 unicast period 500
```

This example shows how to enable Label Consistency Checker (LCC) background scan with a period of 500 milliseconds between buffers in scans for IPv6 unicast tables:

```
lcc ipv6 unicast period 500
```

Enabling RCC and LCC On-demand Scan: Example

This example shows how to run Route Consistency Checker (RCC) on-demand scan for subnet 10.10.0.0/16 in vrf1:

```
show rcc ipv4 unicast 10.10.0.0/16 vrf vrf 1
```

This example shows how to run Label Consistency Checker (LCC) on-demand scan on all labels for IPv6 prefixes:

```
show lcc ipv6 unicast all
```

Where to Go Next

For additional information on the protocols that interact with RIB, you may want to see the following publications:

- *Implementing MPLS Layer 3 VPNs* in *MPLS Configuration Guide for the Cisco CRS Routers*
- *Implementing BGP* in *Routing Configuration Guide for Cisco CRS Routers*
- *Implementing EIGRP* in *Routing Configuration Guide for Cisco CRS Routers*
- *Implementing IS-IS* in *Routing Configuration Guide for Cisco CRS Routers*
- *Implementing OSPF* in *Routing Configuration Guide for Cisco CRS Routers*
- *Implementing RIP* in *Routing Configuration Guide for Cisco CRS Routers*
- *RIB Commands* in *Routing Command Reference for Cisco CRS Routers*

Additional References

Related Documents

Related Topic	Document Title
Routing Information Base commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>RIB Commands on Cisco IOS XR Software in Routing Command Reference for Cisco CRS Routers</i>

Standards and RFCs

Standard/RFC	Title
Draft-ietf-rtgwg-ipfrr-framework-06.txt	<i>IP Fast Reroute Framework</i> , by M. Shand and S. Bryant
Draft-ietf-rtgwg-lf-conv-frmwk-00.txt	<i>A Framework for Loop-free Convergence</i> , by M. Shand and S. Bryant
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

MIBs

MIB	MIBs Link
—	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

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



CHAPTER 10

Implementing RIP

The Routing Information Protocol (RIP) is a classic distance vector Interior Gateway Protocol (IGP) designed to exchange information within an autonomous system (AS) of a small network.

This module describes the concepts and tasks to implement basic RIP routing. Cisco IOS XR software supports a standard implementation of RIP Version 2 (RIPv2) that supports backward compatibility with RIP Version 1 (RIPv1) as specified by RFC 2453.

For RIP configuration information related to the following features, see the [Related Documents, on page 489](#) section of this module.

- Multiprotocol Label Switching (MPLS) Layer 3 Virtual Private Network (VPN)
- Site of Origin (SoO) Support



Note For more information about RIP on the Cisco IOS XR software and complete descriptions of the RIP commands listed in this module, see the [Related Documents, on page 489](#) section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the *Cisco IOS XR Commands Master List for the Cisco CRS Router*.

Feature History for Implementing RIP

Release	Modification
Release 3.3.0	This feature was introduced.
Release 3.5.0	Four-byte autonomous system (AS) number support was added.
Release 4.0.0	MD5 Authentication Using Keychain feature was added.

- [Prerequisites for Implementing RIP, on page 472](#)
- [Information About Implementing RIP, on page 472](#)
- [How to Implement RIP, on page 477](#)
- [Configuration Examples for Implementing RIP, on page 486](#)
- [Additional References, on page 489](#)

Prerequisites for Implementing RIP

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.

Information About Implementing RIP

RIP Functional Overview

RIP Version 1 (RIP v1) is a classful, distance-vector protocol that is considered the easiest routing protocol to implement. Unlike OSPF, RIP broadcasts User Datagram Protocol (UDP) data packets to exchange routing information in internetworks that are flat rather than hierarchical. Network complexity and network management time is reduced. However, as a classful routing protocol, RIP v1 allows only contiguous blocks of hosts, subnets or networks to be represented by a single route, severely limiting its usefulness.

RIP v2 allows more information carried in RIP update packets, such as support for:

- Route summarization
- Classless interdomain routing (CIDR)
- Variable-length subnet masks (VLSMs)
- Autonomous systems and the use of redistribution
- Multicast address 224.0.0.9 for RIP advertisements

The metric that RIP uses to rate the value of different routes is *hop count*. The hop count is the number of routers that can be traversed in a route. A directly connected network has a metric of zero; an unreachable network has a metric of 16. This small range of metrics makes RIP an unsuitable routing protocol for large networks.

Routing information updates are advertised every 30 seconds by default, and new updates discovered from neighbor routers are stored in a routing table.

Only RIP Version 2 (RIP v2), as specified in RFC 2453, is supported on Cisco IOS XR software and, by default, the software only sends and receives RIP v2 packets. However, you can configure the software to send, or receive, or both, only Version 1 packets or only Version 2 packets or both version type packets per interface.

Here are some good reasons to use RIP:

- Compatible with diverse network devices
- Best for small networks, because there is very little overhead, in terms of bandwidth used, configuration, and management time
- Support for legacy host systems

Because of RIP's ease of use, it is implemented in networks worldwide.



Note VRF does not allow configuration of a group applied directly under router RIP. A group can be configured if it is applied globally or under VRF.

Split Horizon for RIP

Normally, routers that are connected to broadcast-type IP networks and that use distance-vector routing protocols employ the *split horizon* mechanism to reduce the possibility of routing loops. Split horizon blocks information about routes from being advertised by a router out of any interface from which that information originated. This behavior usually optimizes communications among multiple routers, particularly when links are broken.

If an interface is configured with secondary IP addresses and split horizon is enabled, updates might not be sourced by every secondary address. One routing update is sourced per network number unless split horizon is disabled.



Note The split horizon feature is enabled by default. In general, we recommend that you do not change the default state of split horizon unless you are certain that your operation requires the change in order to properly advertise routes.

Route Timers for RIP

RIP uses several timers that determine such variables as the frequency of routing updates, the length of time before a route becomes invalid, and other parameters. You can adjust these timers to tune routing protocol performance to better suit your internetwork needs, by making the following timer adjustments to:

- The rate (time in seconds between updates) at which routing updates are sent
- The interval of time (in seconds) after which a route is declared invalid
- The interval (in seconds) during which routing information regarding better paths is suppressed
- The amount of time (in seconds) that must pass before a route is removed from the RIP topology table
- The amount of time delay between RIP update packets

The first four timer adjustments are configurable by the **timers basic** command. The **output-delay** command changes the amount of time delay between RIP update packets. See [Customizing RIP, on page 479](#) for configuration details.

It also is possible to tune the IP routing support in the software to enable faster convergence of the various IP routing algorithms and quickly drop back to redundant routers, if necessary. The total result is to minimize disruptions to end users of the network in situations in which quick recovery is essential.

Route Redistribution for RIP

Redistribution is a feature that allows different routing domains, to exchange routing information. Networking devices that route between different routing domains are called *boundary routers*, and it is these devices that

inject the routes from one routing protocol into another. Routers within a routing domain only have knowledge of routes internal to the domain unless route redistribution is implemented on the boundary routers.

When running RIP in your routing domain, you might find it necessary to use multiple routing protocols within your internetwork and redistribute routes between them. Some common reasons are:

- To advertise routes from other protocols into RIP, such as static, connected, OSPF, and BGP.
- To migrate from RIP to a new Interior Gateway Protocol (IGP) such as EIGRP.
- To retain routing protocol on some routers to support host systems, but upgrade routers for other department groups.
- To communicate among a mixed-router vendor environment. Basically, you might use a protocol specific to Cisco in one portion of your network and use RIP to communicate with devices other than Cisco devices.

Further, route redistribution gives a company the ability to run different routing protocols in work groups or areas in which each is particularly effective. By not restricting customers to using only a single routing protocol, Cisco IOS XR route redistribution is a powerful feature that minimizes cost, while maximizing technical advantage through diversity.

When it comes to implementing route redistribution in your internetwork, it can be very simple or very complex. An example of a simple one-way redistribution is to log into a router on which RIP is enabled and use the **redistribute static** command to advertise only the static connections to the backbone network to pass through the RIP network. For complex cases in which you must consider routing loops, incompatible routing information, and inconsistent convergence time, you must determine why these problems occur by examining how Cisco routers select the best path when more than one routing protocol is running administrative cost.

Default Administrative Distances for RIP

Administrative distance is used as a measure of the trustworthiness of the source of the IP routing information. When a dynamic routing protocol such as RIP is configured, and you want to use the redistribution feature to exchange routing information, it is important to know the default administrative distances for other route sources so that you can set the appropriate distance weight.

This table lists the Default Administrative Distances of Routing Protocols.

Table 7: Default Administrative Distances of Routing Protocols

Routing Protocols	Administrative Distance Value
Connected interface	0
Static route out an interface	0
Static route to next hop	1
EIGRP Summary Route	5
External BGP	20
Internal EIGRP	90

Routing Protocols	Administrative Distance Value
OSPF	110
IS-IS	115
RIP version 1 and 2	120
External EIGRP	170
Internal BGP	200
Unknown	255

An administrative distance is an integer from 0 to 255. In general, the higher the value, the lower the trust rating. An administrative distance of 255 means the routing information source cannot be trusted at all and should be ignored. Administrative distance values are subjective; there is no quantitative method for choosing them.

Routing Policy Options for RIP

Route policies comprise series of statements and expressions that are bracketed with the **route-policy** and **end-policy** keywords. Rather than a collection of individual commands (one for each line), the statements within a route policy have context relative to each other. Thus, instead of each line being an individual command, each policy or set is an independent configuration object that can be used, entered, and manipulated as a unit.

Each line of a policy configuration is a logical subunit. At least one new line must follow the **then**, **else**, and **end-policy** keywords. A new line must also follow the closing parenthesis of a parameter list and the name string in a reference to an AS path set, community set, extended community set, or prefix set. At least one new line must precede the definition of a route policy, AS path set, community set, extended community set, or prefix set. One or more new lines can follow an action statement. One or more new lines can follow a comma separator in a named AS path set, community set, extended community set, or prefix set. A new line must appear at the end of a logical unit of policy expression and may not appear anywhere else.

Authentication Using Keychain in RIP

Authentication using keychain in Cisco IOS XR Routing Information Protocol (RIP) provides mechanism to authenticate all RIP protocol traffic on RIP interface, based keychain authentication. This mechanism uses the Cisco IOS XR security keychain infrastructure to store and retrieve secret keys and use it to authenticate in-bound and out-going traffic on per-interface basis.

Keychain management is a common method of authentication to configure shared secrets on all entities that exchange secrets such as keys, before establishing trust with each other. Routing protocols and network management applications on Cisco IOS XR software often use authentication to enhance security while communicating with peers.



Tip The Cisco IOS XR software system security component implements various system security features including keychain management. Refer these documents for detailed information on keychain management concepts, configuration tasks, examples, and command used to configure keychain management.

- *Implementing Keychain Management* module in *System Security Configuration Guide for Cisco CRS Routers*
- *Keychain Management Commands* module in *System Security Command Reference for Cisco CRS Routers*



Note The keychain by itself has no relevance; therefore, it must be used by an application that needs to communicate by using the keys (for authentication) with its peers. The keychain provides a secure mechanism to handle the keys and rollover based on the lifetime. The Cisco IOS XR keychain infrastructure takes care of the hit-less rollover of the secret keys in the keychain.

Once you have configured a keychain in the IOS XR keychain database and if the same has been configured on a particular RIP interface, it will be used for authenticating all incoming and outgoing RIP traffic on that interface. Unless an authentication keychain is configured on a RIP interface (on the default VRF or a non-default VRF), all RIP traffic will be assumed to be authentic and authentication mechanisms for in-bound RIP traffic and out-bound RIP traffic will be not be employed to secure it.

RIP employs two modes of authentication: keyed message digest mode and clear text mode. Use the **authentication keychain** *keychain-name* **mode** {**md5** | **text**} command to configure authentication using the keychain mechanism.

In cases where a keychain has been configured on RIP interface but the keychain is actually not configured in the keychain database or keychain is not configured with MD5 cryptographic algorithm, all incoming RIP packets on the interface will be dropped. Outgoing packets will be sent without any authentication data.

In-bound RIP Traffic on an Interface

These are the verification criteria for all in-bound RIP packets on a RIP interface when the interface is configured with a keychain.

If...	Then...
The keychain configured on the RIP interface does not exist in the keychain database...	The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.
The keychain is not configured with a MD5 cryptographic algorithm...	The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.
The Address Family Identifier of the first (and only the first) entry in the message is not 0xFFFF, then authentication is not in use...	The packet will be dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.

If...	Then...
The MD5 digest in the 'Authentication Data' is found to be invalid...	The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.
Else, the packet is forwarded for the rest of the processing.	

Out-bound RIP Traffic on an Interface

These are the verification criteria for all out-bound RIP packets on a RIP interface when the interface is configured with a keychain.

If...	Then
The keychain configured on the RIP interface exists in the keychain database ...	The RIP packet passes authentication check at the remote/peer end, provided the remote router is also configured to authenticate the packets using the same keychain.
The keychain is configured with a MD5 cryptographic algorithm...	The RIP packet passes authentication check at the remote/peer end, provided the remote router is also configured to authenticate the packets using the same keychain.
Else, RIP packets fail authentication check.	

How to Implement RIP

This section contains instructions for the following tasks:



Note To save configuration changes, you must commit changes when the system prompts you.

Enabling RIP

This task enables RIP routing and establishes a RIP routing process.

Before you begin

Although you can configure RIP before you configure an IP address, no RIP routing occurs until at least one IP address is configured.

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **neighbor** *ip-address*
4. **broadcast-for-v2**

5. `interface type interface-path-id`
6. `receive version { 1 | 2 | 1 2 }`
7. `send version { 1 | 2 | 1 2 }`
8. `commit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	<code>router rip</code> Example: <pre>RP/0/RP0/CPU0:router(config)# router rip</pre>	Configures a RIP routing process.
Step 3	<code>neighbor ip-address</code> Example: <pre>RP/0/RP0/CPU0:router(config-rip)# neighbor 172.160.1.2</pre>	(Optional) Defines a neighboring router with which to exchange RIP protocol information.
Step 4	<code>broadcast-for-v2</code> Example: <pre>RP/0/RP0/CPU0:router(config-rip)# broadcast-for-v2</pre>	(Optional) Configures RIP to send only Version 2 packets to the broadcast IP address rather than the RIP v2 multicast address (224.0.0.9). This command can be applied at the interface or global configuration level.
Step 5	<code>interface type interface-path-id</code> Example: <pre>RP/0/RP0/CPU0:router(config-rip)# interface GigabitEthernet 0/1/0/0</pre>	(Optional) Defines the interfaces on which the RIP routing protocol runs.
Step 6	<code>receive version { 1 2 1 2 }</code> Example: <pre>RP/0/RP0/CPU0:router(config-rip-if)# receive version 1 2</pre>	(Optional) Configures an interface to accept packets that are: <ul style="list-style-type: none"> • Only RIP v1 • Only RIP v2 • Both RIP v1 and RIP v2
Step 7	<code>send version { 1 2 1 2 }</code> Example: <pre>RP/0/RP0/CPU0:router(config-rip-if)# send version 1 2</pre>	(Optional) Configures an interface to send packets that are: <ul style="list-style-type: none"> • Only RIP v1 • Only RIP v2 • Both RIP v1 and RIP v2
Step 8	<code>commit</code>	

Customizing RIP

This task describes how to customize RIP for network timing and the acceptance of route entries.

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **auto-summary**
4. **timers basic** *update invalid holddown flush*
5. **output-delay** *delay*
6. **nsf**
7. **interface** *type interface-path-id*
8. **metric-zero-accept**
9. **split-horizon** *disable*
10. **poison-reverse**
11. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router rip Example: RP/0/RP0/CPU0:router(config)# router rip	Configures a RIP routing process.
Step 3	auto-summary Example: RP/0/RP0/CPU0:router(config-rip)# auto-summary	(Optional) Enables automatic route summarization of subnet routes into network-level routes. • By default, auto-summary is disabled. Note If you have disconnected subnets, use the no keyword to disable automatic route summarization and permit software to send subnet and host routing information across classful network boundaries.
Step 4	timers basic <i>update invalid holddown flush</i> Example: RP/0/RP0/CPU0:router(config-rip)# timers basic 5 15 15 30	(Optional) Adjusts RIP network timers. Note To view the current and default timer values, view output from the show rip command.
Step 5	output-delay <i>delay</i> Example:	(Optional) Changes the interpacket delay for the RIP updates sent.

	Command or Action	Purpose
	<code>RP/0/RP0/CPU0:router(config-rip)# output-delay 10</code>	Note Use this command if you have a high-end router sending at high speed to a low-speed router that might not be able to receive at that fast a rate.
Step 6	nsf Example: <code>RP/0/RP0/CPU0:router(config-rip)# nsf</code>	(Optional) Configures NSF on RIP routes after a RIP process shutdown or restart.
Step 7	interface <i>type interface-path-id</i> Example: <code>RP/0/RP0/CPU0:router(config-rip)# interface GigabitEthernet 0/1/0/0</code>	(Optional) Defines the interfaces on which the RIP routing protocol runs.
Step 8	metric-zero-accept Example: <code>RP/0/RP0/CPU0:router(config-rip-if)# metric-zero-accept</code>	(Optional) Allows the networking device to accept route entries received in update packets with a metric of zero (0). The received route entry is set to a metric of one (1).
Step 9	split-horizon disable Example: <code>RP/0/RP0/CPU0:router(config-rip-if)# split-horizon disable</code>	(Optional) Disables the split horizon mechanism. <ul style="list-style-type: none"> • By default, split horizon is enabled. • In general, we do not recommend changing the state of the default for the split-horizon command, unless you are certain that your application requires a change to properly advertise routes. If split horizon is disabled on a serial interface (and that interface is attached to a packet-switched network), you must disable split horizon for all networking devices in any relevant multicast groups on that network.
Step 10	poison-reverse Example: <code>RP/0/RP0/CPU0:router(config-rip-if)# poison-reverse</code>	Enables poison reverse processing of RIP router updates.
Step 11	commit	

Control Routing Information

This task describes how to control or prevent routing update exchange and propagation.

Some reasons to control or prevent routing updates are:

- To slow or stop the update traffic on a WAN link—If you do not control update traffic on an on-demand WAN link, the link remains up constantly. By default, RIP routing updates occur every 30 seconds.

- To prevent routing loops—If you have redundant paths or are redistributing routes into another routing domain, you may want to filter the propagation of one of the paths.
- To filter network received in updates — If you do not want other routers from learning a particular device’s interpretation of one or more routes, you can suppress that information.
- To prevent other routers from processing routes dynamically— If you do not want to process routing updates entering the interface, you can suppress that information.
- To preserve bandwidth—You can ensure maximum bandwidth availability for data traffic by reducing unnecessary routing update traffic.

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **neighbor** *ip-address*
4. **interface** *type interface-path-id*
5. **passive-interface**
6. **exit**
7. **interface** *type interface-path-id*
8. **route-policy** { **in** | **out** }
9. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router rip Example: RP/0/RP0/CPU0:router(config)# router rip	Configures a RIP routing process.
Step 3	neighbor <i>ip-address</i> Example: RP/0/RP0/CPU0:router(config-rip)# neighbor 172.160.1.2	(Optional) Defines a neighboring router with which to exchange RIP protocol information.
Step 4	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-rip)# interface GigabitEthernet 0/1/0/0	(Optional) Defines the interfaces on which the RIP routing protocol runs.
Step 5	passive-interface Example:	(Optional) Suppresses the sending of RIP updates on an interface, but not to explicitly configured neighbors.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-rip-if)# passive-interface	
Step 6	exit Example: RP/0/ RP0/CPU0:router(config-rip-if)# exit	(Optional) Returns the router to the next higher configuration mode.
Step 7	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-rip)# interface GigabitEthernet 0/2/0/0	(Optional) Defines the interfaces on which the RIP routing protocol runs.
Step 8	route-policy { in out } Example: RP/0/RP0/CPU0:router(config-rip-if)# route-policy out	(Optional) Applies a routing policy to updates advertised to or received from a RIP neighbor.
Step 9	commit	

Creating a Route Policy for RIP

This task defines a route policy and shows how to attach it to an instance of a RIP process. Route policies can be used to:

- Control routes sent and received
- Control which routes are redistributed
- Control origination of the default route

A route policy definition consists of the **route-policy** command and *name* argument followed by a sequence of optional policy statements, and then closes with the **end-policy** command.

A route policy is not useful until it is applied to routes of a routing protocol.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **set rip-metric** *number*
4. **end-policy**
5. **commit**
6. **configure**
7. **router rip**
8. **route-policy** *route-policy-name* { **in** | **out** }
9. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	<code>route-policy name</code> Example: RP/0/RP0/CPU0:router(config)# route-policy IN-IPv4	Defines a route policy and enters route-policy configuration mode.
Step 3	<code>set rip-metric number</code> Example: RP/0/RP0/CPU0:router(config-rpl)# set rip metric 42	(Optional) Sets the RIP metric attribute.
Step 4	<code>end-policy</code> Example: RP/0/RP0/CPU0:router(config-rpl)# end-policy	Ends the definition of a route policy and exits route-policy configuration mode.
Step 5	<code>commit</code>	
Step 6	<code>configure</code>	
Step 7	<code>router rip</code> Example: RP/0/RP0/CPU0:router(config)# router rip	Configures a RIP routing process.
Step 8	<code>route-policy route-policy-name { in out }</code> Example: RP/0/RP0/CPU0:router(config-rip)# route-policy rpl in	Applies a routing policy to updates advertised to or received from an RIP neighbor.
Step 9	<code>commit</code>	

Configuring RIP Authentication Keychain

Configuring RIP Authentication Keychain for IPv4 Interface on a Non-default VRF

Perform this task to configure a RIP authentication keychain for IPv4 interface on a non-default VRF.

Before you begin

All keychains need to be configured in Cisco IOS XR keychain database using configuration commands described in *Implementing Keychain Management* module of *System Security Configuration Guide for Cisco CRS Routers* before they can be applied to a RIP interface/VRF.

The **authentication keychain** *keychain-name* and **mode md5** configurations will accept the name of a keychain that has not been configured yet in the IOS XR keychain database or a keychain that has been configured in IOS XR keychain database without MD5 cryptographic algorithm. However, in both these cases, all incoming packets on the interface will be dropped and outgoing packets will be sent without authentication data.

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **vrf vrf_name**
4. **interface type interface-path-id**
5. Use one of these commands:
 - **authentication keychain** *keychain-name* **mode md5**
 - **authentication keychain** *keychain-name* **mode text**
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router rip Example: RP/0/RP0/CPU0:router(config)#router rip	Configures a RIP routing process.
Step 3	vrf vrf_name Example: RP/0/RP0/CPU0:router(config-rip)#vrf vrf_rip_auth	Configures a non-default VRF
Step 4	interface type interface-path-id Example: RP/0/RP0/CPU0:router(config-rip-vrf)#interface POS 0/6/0/0	Defines the interface on which the RIP routing protocol runs.
Step 5	Use one of these commands: <ul style="list-style-type: none"> • authentication keychain <i>keychain-name</i> mode md5 • authentication keychain <i>keychain-name</i> mode text Example: RP/0/RP0/CPU0:router(config-rip-if)#authentication keychain key1 mode md5 Or RP/0/RP0/CPU0:router(config-rip-if)#authentication keychain key1 mode text	Configures an authentication keychain mode for RIP. <ul style="list-style-type: none"> • md5—Keyed message digest (md5) authentication mode • text—Clear text authentication mode
Step 6	commit	

Configuring RIP Authentication Keychain for IPv4 Interface on Default VRF

Perform this task to configure a RIP authentication keychain for IPv4 interface (on the default VRF).

Before you begin

All keychains need to be configured in Cisco IOS XR keychain database using configuration commands described in *Implementing Keychain Management* module of *System Security Configuration Guide for Cisco CRS Routers* before they can be applied to a RIP interface/VRF.

The **authentication keychain** *keychain-name* and **mode md5** configurations will accept the name of a keychain that has not been configured yet in the IOS XR keychain database or a keychain that has been configured in IOS XR keychain database without MD5 cryptographic algorithm. However, in both these cases, all incoming packets on the interface will be dropped and outgoing packets will be sent without authentication data.

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **interface** *type interface-path-id*
4. Use one of these commands:
 - **authentication keychain** *keychain-name* **mode md5**
 - **authentication keychain** *keychain-name* **mode text**
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router rip Example: RP/0/RP0/CPU0:router(config)#router rip	Configures a RIP routing process.
Step 3	interface <i>type interface-path-id</i> Example: RP/0/RP0/CPU0:router(config-rip)#interface POS 0/6/0/0	Defines the interface on which the RIP routing protocol runs.
Step 4	Use one of these commands: <ul style="list-style-type: none"> • authentication keychain <i>keychain-name</i> mode md5 • authentication keychain <i>keychain-name</i> mode text Example: RP/0/RP0/CPU0:router(config-rip-if)#authentication keychain key1 mode md5 Or	Configures an authentication keychain mode for RIP. <ul style="list-style-type: none"> • md5—Keyed message digest (md5) authentication mode • text—Clear text authentication mode

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-rip-if)#authentication keychain key1 mode text	
Step 5	commit	

Configuration Examples for Implementing RIP

This section provides the following configuration examples:

Configuring a Basic RIP Configuration: Example

The following example shows two Gigabit Ethernet interfaces configured with RIP.

```
interface GigabitEthernet0/6/0/0
  ipv4 address 172.16.0.1 255.255.255.0
  !
interface GigabitEthernet0/6/0/2
  ipv4 address 172.16.2.12 255.255.255.0
  !
router rip
  interface GigabitEthernet0/6/0/0
  !
  interface GigabitEthernet0/6/0/2
  !
  !
```

Configuring RIP on the Provider Edge: Example

The following example shows how to configure basic RIP on the PE with two VPN routing and forwarding (VRF) instances.

```
router rip
  interface GigabitEthernet0/6/0/0
  !
  vrf vpn0
    interface GigabitEthernet0/6/0/2
    !
  !
  vrf vpn1
    interface GigabitEthernet0/6/0/3
    !
  !
  !
```

Adjusting RIP Timers for each VRF Instance: Example

The following example shows how to adjust RIP timers for each VPN routing and forwarding (VRF) instance.

For VRF instance vpn0, the **timers basic** command sets updates to be broadcast every 10 seconds. If a router is not heard from in 30 seconds, the route is declared unusable. Further information is suppressed for an additional 30 seconds. At the end of the flush period (45 seconds), the route is flushed from the routing table.

For VRF instance vpn1, timers are adjusted differently: 20, 60, 60, and 70 seconds.

The **output-delay** command changes the interpacket delay for RIP updates to 10 milliseconds on vpn1. The default is that interpacket delay is turned off.

```
router rip
 interface GigabitEthernet0/6/0/0
 !
 vrf vpn0
  interface GigabitEthernet0/6/0/2
  !
  timers basic 10 30 30 45
 !
 vrf vpn1
  interface GigabitEthernet0/6/0/3
  !
  timers basic 20 60 60 70
  output-delay 10
 !
 !
```

Configuring Redistribution for RIP: Example

The following example shows how to redistribute Border Gateway Protocol (BGP) and static routes into RIP.

The RIP metric used for redistributed routes is determined by the route policy. If a route policy is not configured or the route policy does not set RIP metric, the metric is determined based on the redistributed protocol. For VPNv4 routes redistributed by BGP, the RIP metric set at the remote PE router is used, if valid.

In all other cases (BGP, IS-IS, OSPF, EIGRP, connected, static), the metric set by the **default-metric** command is used. If a valid metric cannot be determined, then redistribution does not happen.

```
route-policy ripred
 set rip-metric 5
end-policy
!

router rip
 vrf vpn0
  interface GigabitEthernet0/6/0/2
  !
  redistribute connected
  default-metric 3
 !
 vrf vpn1
  interface GigabitEthernet0/6/0/3
  !
  redistribute bgp 100 route-policy ripred
  redistribute static
  default-metric 3
 !
 !
```

Configuring Route Policies for RIP: Example

The following example shows how to configure inbound and outbound route policies that are used to control which route updates are received by a RIP interface or sent out from a RIP interface.

```

prefix-set pf1
 10.1.0.0/24
end-set
!

prefix-set pf2
 150.10.1.0/24
end-set
!

route-policy policy_in
 if destination in pf1 then
  pass
 endif
end-policy
!

route-policy pass-all
 pass
end-policy
!

route-policy infil
 if destination in pf2 then
  add rip-metric 2
  pass
 endif
end-policy
!

router rip
 interface GigabitEthernet0/6/0/0
  route-policy policy_in in
  !
 interface GigabitEthernet0/6/0/2
  !
  route-policy infil in
  route-policy pass-all out

```

Configuring Passive Interfaces and Explicit Neighbors for RIP: Example

The following example shows how to configure passive interfaces and explicit neighbors. When an interface is passive, it only accepts routing updates. In other words, no updates are sent out of an interface except to neighbors configured explicitly.

```

router rip
 interface GigabitEthernet0/6/0/0
  passive-interface
  !
 interface GigabitEthernet0/6/0/2
  !
  neighbor 172.17.0.1
  neighbor 172.18.0.5

```

!

Controlling RIP Routes: Example

The following example shows how to use the **distance** command to install RIP routes in the Routing Information Base (RIB). The **maximum-paths** command controls the number of maximum paths allowed per RIP route.

```
router rip
 interface GigabitEthernet0/6/0/0
  route-policy polin in
  !
  distance 110
  maximum-paths 8
  !
```

Configuring RIP Authentication Keychain: Example

This example shows how to apply an authentication keychain on a RIP default VRF interface:

```
router rip
 interface POS0/6/0/0
  authentication keychain key1 mode md5
  !
  !
end
```

This example shows how to apply an authentication keychain on a RIP non-default interface:

```
router rip
 vrf rip_keychain_vrf
  interface POS0/6/0/0
    authentication keychain key1 mode md5
  !
  !
end
```

Additional References

The following sections provide references related to implementing RIP.

Related Documents

Related Topic	Document Title
RIP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Routing Command Reference for Cisco CRS Routers</i>

Related Topic	Document Title
MPLS VPN support for RIP feature information	<i>Implementing MPLS Traffic Engineering on Cisco IOS XR Software</i> module in the <i>MPLS Configuration Guide for the Cisco CRS Routers</i>
Site of Origin (SoO) support for RIP feature information	<i>Implementing MPLS Traffic Engineering on Cisco IOS XR Software</i> module in the <i>MPLS Configuration Guide for the Cisco CRS Routers</i>
Cisco IOS XR getting started documentation	<i>Cisco IOS XR Getting Started Guide for the Cisco CRS Router</i>
Information about user groups and task IDs	<i>Configuring AAA Services on Cisco IOS-XR Software</i> module in the <i>System Security Configuration Guide for Cisco CRS Routers</i>

Standards

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

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

RFCs

RFCs	Title
RFC 2453	RIP Version 2

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport



CHAPTER 11

Implementing Routing Policy

A routing 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.

This module describes how routing protocols make decisions to advertise, aggregate, discard, distribute, export, hold, import, redistribute and modify the routes based on configured routing policy.

The routing policy language (RPL) provides a single, straightforward language in which all routing policy needs can be expressed. RPL was designed to support large-scale routing configurations. It greatly reduces the redundancy inherent in previous routing policy configuration methods. RPL streamlines the routing policy configuration, reduces system resources required to store and process these configurations, and simplifies troubleshooting.



Note For more information about routing policy on the Cisco IOS XR software and complete descriptions of the routing policy commands listed in this module, see the [Related Documents, on page 580](#) section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the *Cisco IOS XR Commands Master List for the Cisco CRS Router*.

Feature History for Implementing Routing Policy

Release	Modification
Release 2.0	This feature was introduced.
Release 3.0	No modification.
Release 3.2	No modification.
Release 3.3.0	Support was added for EIGRP, RIP policy, clear-policy, debug, OSPF area-in, and OSPF area-out attach points.
Release 3.4.0	Support was added for the BGP next-hop attach point. Support was also added for null sets and global parameterization.

Release	Modification
Release 3.5.0	The following features were added: <ul style="list-style-type: none"> • GNU Nano text editor for editing RPL policies. • Enhanced prefix match functionality. • Parameterization at attach points • New 'done' disposition policy statement type
Release 3.6.0	No modification.
Release 3.7.0	Support was added for IS-IS inter-area-propagate attach point, OSPF spf-prefix-priority attach point, and PIM policy rpf-topology attach point.
Release 3.8.0	The vrf <i>vrf-name</i> keyword and argument were added for the PIM policy rpf-topology attach point to support MVPN extranet. The prefix matching performance feature was enhanced to use Radix trie lookup.
Release 3.9.0	Parameterization was supported at all attach points.
Release 4.2.0	The following features were added: <ul style="list-style-type: none"> • Hierarchical Conditions • Apply Condition Policies
Release 4.2.1	The following features were introduced: <ul style="list-style-type: none"> • Enhanced Prefix-length Manipulation. • Nested Wildcard Apply Policy. • Editing Routing Policy Language set elements Using XML. • Support 'set' as a valid operator for the 'med' attribute at the bgp export and bgp import attach points.
Release 4.3.1	The following features were introduced: <ul style="list-style-type: none"> • VRF RPL Based Import Policy • Flexible L3VPN Label Allocation

- [Prerequisites for Implementing Routing Policy, on page 493](#)
- [Restrictions for Implementing Routing Policy, on page 493](#)
- [Information About Implementing Routing Policy, on page 493](#)
- [How to Implement Routing Policy, on page 570](#)
- [Configuration Examples for Implementing Routing Policy, on page 573](#)
- [Additional References, on page 580](#)

Prerequisites for Implementing Routing Policy

The following are prerequisites for implementing Routing Policy on Cisco IOS XR Software:

- 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.
- Border Gateway Protocol (BGP), integrated Intermediate System-to-Intermediate System (IS-IS), or Open Shortest Path First (OSPF) must be configured in your network.

Restrictions for Implementing Routing Policy

These restrictions apply when working with Routing Policy Language implementation on Cisco IOS XR software:

- An individual policy definition of up to 1000 statements are supported. The total number of statements within a policy can be extended to 4000 statements using hierarchical policy constructs. However, this limit is restricted with the use of **apply** statements.
- When a policy that is attached directly or indirectly to an attach point needs to be modified, a single **commit** operation cannot be performed when:
 - Removing a set or policy referred by another policy that is attached to any attach point directly or indirectly.
 - Modifying the policy to remove the reference to the same set or policy that is getting removed.

The **commit** must be performed in two steps:

1. Modify the policy to remove the reference to the policy or set and then **commit**.
 2. Remove the policy or set and **commit**.
- Per-vrf label mode is not supported for Carrier Supporting Carrier (CSC) network with internal and external BGP multipath setup.
 - You cannot change the next hop address to an IPv6 address through RPL policy for a route that starts from an IPv4 peer.

Information About Implementing Routing Policy

To implement RPL, you need to understand the following concepts:

Routing Policy Language

This section contains the following information:

Routing Policy Language Overview

RPL was developed to support large-scale routing configurations. RPL has several fundamental capabilities that differ from those present in configurations oriented to traditional route maps, access lists, and prefix lists. The first of these capabilities is the ability to build policies in a modular form. Common blocks of policy can be defined and maintained independently. These common blocks of policy can then be applied from other blocks of policy to build complete policies. This capability reduces the amount of configuration information that needs to be maintained. In addition, these common blocks of policy can be parameterized. This parameterization allows for policies that share the same structure but differ in the specific values that are set or matched against to be maintained as independent blocks of policy. For example, three policies that are identical in every way except for the local preference value they set can be represented as one common parameterized policy that takes the varying local preference value as a parameter to the policy.

The policy language introduces the notion of sets. Sets are containers of similar data that can be used in route attribute matching and setting operations. Four set types exist: prefix-sets, community-sets, as-path-sets, and extcommunity-sets. These sets hold groupings of IPv4 or IPv6 prefixes, community values, AS path regular expressions, and extended community values, respectively. Sets are simply containers of data. Most sets also have an inline variant. An inline set allows for small enumerations of values to be used directly in a policy rather than having to refer to a named set. Prefix lists, community lists, and AS path lists must be maintained even when only one or two items are in the list. An inline set in RPL allows the user to place small sets of values directly in the policy body without having to refer to a named set.

Decision making, such as accept and deny, is explicitly controlled by the policy definitions themselves. RPL combines matching operators, which may use set data, with the traditional Boolean logic operators AND, OR, and NOT into complex conditional expressions. All matching operations return a true or false result. The execution of these conditional expressions and their associated actions can then be controlled by using simple *if then*, *elseif*, and *else* structures, which allow the evaluation paths through the policy to be fully specified by the user.

Routing Policy Language Structure

This section describes the basic structure of RPL.

Names

The policy language provides two kinds of persistent, namable objects: sets and policies. Definition of these objects is bracketed by beginning and ending command lines. For example, to define a policy named `test`, the configuration syntax would look similar to the following:

```
route-policy test
[ . . . policy statements . . . ]
end-policy
```

Legal names for policy objects can be any sequence of the upper- and lowercase alphabetic characters; the numerals 0 to 9; and the punctuation characters period, hyphen, and underscore. A name must begin with a letter or numeral.

Sets

In this context, the term set is used in its mathematical sense to mean an unordered collection of unique elements. The policy language provides sets as a container for groups of values for matching purposes. Sets

are used in conditional expressions. The elements of the set are separated by commas. Null (empty) sets are allowed.

In the following example:

```
prefix-set backup-routes
  # currently no backup routes are defined
end-set
```

a condition such as:

```
if destination in backup-routes then
```

evaluates as FALSE for every route, because there is no match-condition in the prefix set that it satisfies.

Five kinds of sets exist: [as-path-set, on page 496](#), [community-set, on page 496](#), [extcommunity-set, on page 497](#), [prefix-set, on page 500](#), and [rd-set, on page 502](#). You may want to perform comparisons against a small number of elements, such as two or three community values, for example. To allow for these comparisons, the user can enumerate these values directly. These enumerations are referred to as *inline sets*. Functionally, inline sets are equivalent to named sets, but allow for simple tests to be inline. Thus, comparisons do not require that a separate named set be maintained when only one or two elements are being compared. See the set types described in the following sections for the syntax. In general, the syntax for an inline set is a comma-separated list surrounded by parentheses as follows: (element-entry , element-entry , element-entry, ...element-entry), where element-entry is an entry of an item appropriate to the type of usage such as a prefix or a community value.

The following is an example using an inline community set:

```
route-policy sample-inline
if community matches-any ([10..15]:100) then
set local-preference 100
endif
end-policy
```

The following is an equivalent example using the named set test-communities:

```
community-set test-communities
10:100,
11:100,
12:100,
13:100,
14:100,
15:100
end-set

route-policy sample
if community matches-any test-communities then
set local-preference 100
endif
end-policy
```

Both of these policies are functionally equivalent, but the inline form does not require the configuration of the community set just to store the six values. You can choose the form appropriate to the configuration context. In the following sections, examples of both the named set version and the inline form are provided where appropriate.

as-path-set

An AS path set comprises operations for matching an AS path attribute. The only matching operation is a regular expression match.

Named Set Form

The named set form uses the **ios-regex** keyword to indicate the type of regular expression and requires single quotation marks around the regular expression.

The following is a sample definition of a named AS path set:

```
as-path-set aset1
ios-regex '_42$',
ios-regex '_127$'
end-set
```

This AS path set comprises two elements. When used in a matching operation, this AS path set matches any route whose AS path ends with either the autonomous system (AS) number 42 or 127.

To remove the named AS path set, use the **no as-path-set aset1** command-line interface (CLI) command.



Note Regular expression matching is CPU intensive. The policy performance can be substantially improved by either collapsing the regular expression patterns together to reduce the total number of regular expression invocations or by using equivalent native as-path match operations such as 'as-path neighbor-is', 'as-path originates-from' or 'as-path passes-through'.

Inline Set Form

The inline set form is a parenthesized list of comma-separated expressions, as follows:

```
(ios-regex '_42$', ios-regex '_127$')
```

This set matches the same AS paths as the previously named set, but does not require the extra effort of creating a named set separate from the policy that uses it.

community-set

A community-set holds community values for matching against the BGP community attribute. A community is a 32-bit quantity. Integer community values *must* be split in half and expressed as two unsigned decimal integers in the range from 0 to 65535, separated by a colon. Single 32-bit community values are not allowed. The following is the named set form:

Named Set Form

```
community-set cset1
12:34,
12:56,
12:78,
internet
end-set
```

Inline Set Form

```
(12:34, 12:56, 12:78)
($as:34, $as:$tag1, 12:78, internet)
```

The inline form of a community-set also supports parameterization. Each 16-bit portion of the community may be parameterized. See the [Parameterization, on page 506](#) for more information.

RPL provides symbolic names for the standard well-known community values: internet is 0:0, no-export is 65535:65281, no-advertise is 65535:65282, and local-as is 65535is-empty:65283.

RPL also provides a facility for using *wildcards* in community specifications. A wildcard is specified by inserting an asterisk (*) in place of one of the 16-bit portions of the community specification; the wildcard indicates that any value for that portion of the community matches. Thus, the following policy matches all communities in which the autonomous system part of the community is 123:

```
community-set cset3
123:*
end-set
```

A community set can either be empty, or contain one or more community values. When used with an empty community set, the **is-empty** operator will evaluate to TRUE and the **matches-any** and **matches-every** operators will evaluate to FALSE.

extcommunity-set

An extended community-set is analogous to a community-set except that it contains extended community values instead of regular community values. It also supports named forms and inline forms. There are three types of extended community sets: cost, soo, and rt.

As with community sets, the inline form supports parameterization within parameterized policies. Either portion of the extended community value can be parameterized.

Wildcards (*) and regular expressions are allowed for extended community set elements.

Every extended community-set must contain at least one extended community value. Empty extended community-sets are invalid and rejected.

The following are syntactic examples:

Named Form for Extcommunity-set Cost

A cost set is an extcommunity set used to store cost EIGRP Cost Community type extended community type communities.

```

extcommunity-set cost a_cost_set
    IGP:1:10
end-set

```

These options are supported under extended community set Cost:

```

RP/0/RP0/CPU0:router(config)#extcommunity-set cost cost_set
RP/0/RP0/CPU0:router(config-ext)#?
  #-remark      Remark beginning with '#'
  <0-255>       decimal number
  abort         Discard RPL definition and return to top level config
  end-set       End of set definition
  exit          Exit from this submode
  igp:          Cost Community with IGP as point of insertion
  pre-bestpath: Cost Community with Pre-Bestpath as point of insertion
  show          Show partial RPL configuration

```

Option	Description
#-remark	Remark beginning with '#'
<0-255>	decimal number
abort	Discard RPL definition and return to top level config
end-set	End of set definition
exit	Exit from this submode
igp:	Cost Community with IGP as point of insertion
pre-bestpath:	Cost Community with Pre-Bestpath as point of insertion
show	Show partial RPL configuration

Named Form for Extcommunity-set RT

An rt set is an extcommunity set used to store BGP Route Target (RT) extended community type communities:

```

extcommunity-set rt a_rt_set
    1.2.3.4:666
    1234:666,
    1.2.3.4:777,
    4567:777
end-set

Inline Set Form for Extcommunity-set RT

(1.2.3.4:666, 1234:666, 1.2.3.4:777, 4567:777)
($ipaddr:666, 1234:$tag, 1.2.3.4:777, $tag2:777)

```

These options are supported under extended community set RT:

```

RP/0/RP0/CPU0:router(config)#extcommunity-set rt rt_set
RP/0/RP0/CPU0:router(config-ext)#?
  #-remark      Remark beginning with '#'
  *             Wildcard (any community or part thereof)
  <1-4294967295> 32-bit decimal number

```



```

<1-65535>      16-bit decimal number
A.B.C.D/M:N    Extended community - IPv4 prefix format
A.B.C.D:N      Extended community - IPv4 format
ASN:N          Extended community - ASPLAIN format
X.Y:N         Extended community - ASDOT format
abort          Discard RPL definition and return to top level config
dfa-regex      DFA style regular expression
end-set        End of set definition
exit           Exit from this submode
ios-regex      Traditional IOS style regular expression
show           Show partial RPL configuration

```

Option	Description
#-remark	Remark beginning with '#'
*	Wildcard (any community or part thereof)
<1-4294967295>	32-bit decimal number
<1-65535>	16-bit decimal number
A.B.C.D/M:N	Extended community - IPv4 prefix format
A.B.C.D:N	Extended community - IPv4 format
ASN:N	Extended community - ASPLAIN format
X.Y:N	Extended community - ASDOT format
abort	Discard RPL definition and return to top level config
dfa-regex	DFA style regular expression
end-set	End of set definition
exit	Exit from this submode
ios-regex	Traditional IOS style regular expression
show	Show partial RPL configuration

Named Form for Extcommunity-set Soo

A soo set is an extcommunity set used to store BGP Site-of-Origin (SoO) extended community type communities:

```

extcommunity-set soo a_soo_set
1.1.1:100,
  100:200
end-set

```

These options are supported under extended community set Soo:

```

RP/0/RP0/CPU0:router(config)#extcommunity-set soo soo_set
RP/0/RP0/CPU0:router(config-ext)#?
  #-remark      Remark beginning with '#'
  *             Wildcard (any community or part thereof)
  <1-4294967295> 32-bit decimal number
  <1-65535>      16-bit decimal number
  A.B.C.D/M:N    Extended community - IPv4 prefix format

```

```

A.B.C.D:N      Extended community - IPv4 format
ASN:N         Extended community - ASPLAIN format
X.Y:N         Extended community - ASDOT format
abort         Discard RPL definition and return to top level config
dfa-regex     DFA style regular expression
end-set       End of set definition
exit          Exit from this submode
ios-regex     Traditional IOS style regular expression
show          Show partial RPL configuration

```

Option	Description
#-remark	Remark beginning with '#'
*	Wildcard (any community or part thereof)
<1-4294967295>	32-bit decimal number
<1-65535>	16-bit decimal number
A.B.C.D/M:N	Extended community - IPv4 prefix format
A.B.C.D:N	Extended community - IPv4 format
ASN:N	Extended community - ASPLAIN format
X.Y:N	Extended community - ASDOT format
abort	Discard RPL definition and return to top level config
dfa-regex	DFA style regular expression
end-set	End of set definition
exit	Exit from this submode
ios-regex	Traditional IOS style regular expression
show	Show partial RPL configuration

prefix-set

A prefix-set holds IPv4 or IPv6 prefix match specifications, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. The address is a standard dotted-decimal IPv4 or colon-separated hexadecimal IPv6 address. The mask length, if present, is a nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6) following the address and separated from it by a slash. The optional minimum matching length follows the address and optional mask length and is expressed as the keyword **ge** (mnemonic for **greater than or equal to**), followed by a nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6). The optional maximum matching length follows the rest and is expressed by the keyword **le** (mnemonic for **less than or equal to**), followed by yet another nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6). A syntactic shortcut for specifying an exact length for prefixes to match is the **eq** keyword (mnemonic for **equal to**).

If a prefix match specification has no mask length, then the default mask length is 32 for IPv4 and 128 for IPv6. The default minimum matching length is the mask length. If a minimum matching length is specified, then the default maximum matching length is 32 for IPv4 and 128 for IPv6. Otherwise, if neither minimum nor maximum is specified, the default maximum is the mask length.

Radix trie lookup is used to perform prefix-set matching.

The prefix-set itself is a comma-separated list of prefix match specifications. The following are examples:

```
prefix-set legal-ipv4-prefix-examples
  10.0.1.1,
  10.0.2.0/24,
  10.0.3.0/24 ge 28,
  10.0.4.0/24 le 28,
  10.0.5.0/24 ge 26 le 30,
  10.0.6.0/24 eq 28,
  10.0.7.2/32 ge 16 le 24,
  10.0.8.0/26 ge 8 le 16
end-set

prefix-set legal-ipv6-prefix-examples
  2001:0:0:1::/64,
  2001:0:0:2::/64 ge 96,
  2001:0:0:2::/64 ge 96 le 100,
  2001:0:0:2::/64 eq 100
end-set
```

The first element of the prefix-set matches only one possible value, 10.0.1.1/32 or the host address 10.0.1.1. The second element matches only one possible value, 10.0.2.0/24. The third element matches a range of prefix values, from 10.0.3.0/28 to 10.0.3.255/32. The fourth element matches a range of values, from 10.0.4.0/24 to 10.0.4.240/28. The fifth element matches prefixes in the range from 10.0.5.0/26 to 10.0.5.252/30. The sixth element matches any prefix of length 28 in the range from 10.0.6.0/28 through 10.0.6.240/28. The seventh element matches any prefix of length 32 in the range 10.0.[0..255].2/32 (from 10.0.0.2/32 to 10.0.255.2). The eighth element matches any prefix of length 26 in the range 10.[0..255].8.0/26 (from 10.0.8.0/26 to 10.255.8.0/26).

The following prefix-set consists entirely of invalid prefix match specifications:

```
prefix-set ILLEGAL-PREFIX-EXAMPLES
  10.1.1.1 ge 16,
  10.1.2.1 le 16,
  10.1.3.0/24 le 23,
  10.1.4.0/24 ge 33,
  10.1.5.0/25 ge 29 le 28
end-set
```

Neither the minimum length nor maximum length is valid without a mask length. For IPv4, the minimum length must be less than 32, the maximum length of an IPv4 prefix. For IPv6, the minimum length must be less than 128, the maximum length of an IPv6 prefix. The maximum length must be equal to or greater than the minimum length.

Enhanced Prefix-length Manipulation

The enhanced prefix-length manipulation support in a prefix-set enhances the prefix-range on using **ge** semantics in prefix match specifications. This caters to have a single entry that matches prefixes 0.0.0.0/0, 0.0.0.0/1, 0.0.0.0/2, ..., 0.0.0.0/32. The prefix-length can be manipulated with **ge** semantics as prefix-set (0.0.0.0/30 ge 0 le 32) that will match all prefixes in the range 0.0.0.0/0 to 0.0.0.3/32. With this, the single prefix-set entry 0.0.0.0/32 ge 0 le 32 will match prefixes 0.0.0.0/0, 0.0.0.0/1, 0.0.0.0/2, ..., 0.0.0.0/32.

These are prefix ranges with the IPv4 prefix syntax along with corresponding mask length ranges:

- <A.B.C.D>/<len> ge <G> le <L>

- <A.B.C.D>/[<len>..<G>] (if <len> is lesser than <G>)
- <A.B.C.D>/[<G>..<len>] (if <len> is greater than <G>)
- <A.B.C.D>/<len> ge <G>
 - <A.B.C.D>/[<len>..<G>] (if <len> is lesser than <G>)
 - <A.B.C.D>/[<G>..<len>] (if <len> is greater than <G>)
- <A.B.C.D>/<len> eq <E>
 - <A.B.C.D>/[<len>..<E>] (if <len> is lesser than <E>)
 - <A.B.C.D>/[<E>..<len>] (if <len> is greater than <E>)

ACL Support in RPL Prefix Sets

Access Control List (ACL) type prefix set entries holds IPv4 or IPv6 prefix match specifications, each of which has an address and a wildcard mask. The address and wildcard mask is a standard dotted-decimal IPv4 or colon-separated hexadecimal IPv6 address. The set of bits to be matched are provided in the form of wildcard also called as inverted mask in which a binary 0 means a mandatory match and binary 1 means a do not match condition. The prefix set allows to specify contiguous and non-contiguous set of bits that should be matched in any route.

rd-set

An rd-set is used to create a set with route distinguisher (RD) elements. An RD set is a 64-bit value prepended to an IPv4 address to create a globally unique Border Gateway Protocol (BGP) VPN IPv4 address.

You can define RD values with the following commands:

- *a.b.c.d:m*:*—BGP VPN RD in IPv4 format with a wildcard character. For example, 10.0.0.2:255.255.0.0:*
- *a.b.c.d/m:n*—BGP VPN RD in IPv4 format with a mask. For example, 10.0.0.2:255.255.0.0:666.
- *a.b.c.d:***—BGP VPN RD in IPv4 format with a wildcard character. For example, 10.0.0.2:255.255.0.0.
- *a.b.c.d:n*—BGP VPN RD in IPv4 format. For example, 10.0.0.2:666.
- *asn:**—BGP VPN RD in ASN format with a wildcard character. For example, 10002:255.255.0.0.
- *asn:n*—BGP VPN RD in ASN format. For example, 10002:666.

The following is an example of an rd-set:

```
rd-set rdset1
  10.0.0.0/8:*,
  10.0.0.0/8:777,
  10.0.0.0:*,
  10.0.0.0:777,
  65000:*,
  65000:777
end-set
```

Routing Policy Language Components

Four main components in the routing policy language are involved in defining, modifying, and using policies: the configuration front end, policy repository, execution engine, and policy clients themselves.

The configuration front end (CLI) is the mechanism to define and modify policies. This configuration is then stored on the router using the normal storage means and can be displayed using the normal configuration **show** commands.

The second component of the policy infrastructure, the policy repository, has several responsibilities. First, it compiles the user-entered configuration into a form that the execution engine can understand. Second, it performs much of the verification of policies; and it ensures that defined policies can actually be executed properly. Third, it tracks which attach points are using which policies so that when policies are modified the appropriate clients are properly updated with the new policies relevant to them.

The third component is the execution engine. This component is the piece that actually runs policies as the clients request. The process can be thought of as receiving a route from one of the policy clients and then executing the actual policy against the specific route data.

The fourth component is the policy clients (the routing protocols). This component calls the execution engine at the appropriate times to have a given policy be applied to a given route, and then perform some number of actions. These actions may include deleting the route if policy indicated that it should be dropped, passing along the route to the protocol decision tree as a candidate for the best route, or advertising a policy modified route to a neighbor or peer as appropriate.

Routing Policy Language Usage

This section provides basic routing policy language usage examples. See the [How to Implement Routing Policy, on page 570](#) for detailed information on how to implement routing policy language.

Pass Policy

The following example shows how the policy accepts all presented routes without modifying the routes.

```
route-policy quickstart-pass
pass
end-policy
```

Drop Everything Policy

The following example shows how the policy explicitly rejects all routes presented to it. This type of policy is used to ignore everything coming from a specific peer.

```
route-policy quickstart-drop
drop
end-policy
```

Ignore Routes with Specific AS Numbers in the Path

The following example shows the policy definition in three parts. First, the **as-path-set** command defines three regular expressions to match against an AS path. Second, the **route-policy** command applies the AS path set to a route. If the AS path attribute of the route matches the regular expression defined with the

as-path-set command, the protocol refuses the route. Third, the route policy is attached to BGP neighbor 10.0.1.2. BGP consults the policy named `ignore_path_as` on routes received (imported) from neighbor 10.0.1.2.

```
as-path-set ignore_path
ios-regex '_11_',
ios-regex '_22_',
ios-regex '_33_'
end-set

route-policy ignore_path_as
if as-path in ignore_path then
drop
else
pass
endif
end-policy

router bgp 2
neighbor 10.0.1.2 address-family ipv4 unicast policy ignore_path_as in
```

Set Community Based on MED

The following example shows how the policy tests the MED of a route and modifies the community attribute of the route based on the value of the MED. If the MED value is 127, the policy adds the community 123:456 to the route. If the MED value is 63, the policy adds the value 123:789 to the community attribute of the route. Otherwise, the policy removes the community 123:123 from the route. In any case, the policy instructs the protocol to accept the route.

```
route-policy quickstart-med
if med eq 127 then
set community (123:456) additive
elseif med eq 63 then
set community (123:789) additive
else
delete community in (123:123)
endif
pass
end-policy
```

Set Local Preference Based on Community

The following example shows how the community-set named `quickstart-communities` defines community values. The route policy named `quickstart-localpref` tests a route for the presence of the communities specified in the `quickstart-communities` community set. If any of the community values are present in the route, the route policy sets the local preference attribute of the route to 31. In any case, the policy instructs the protocol to accept the route.

```
community-set quickstart-communities
987:654,
987:543,
987:321,
987:210
end-set

route-policy quickstart-localpref
if community matches-any quickstart-communities then
set local-preference 31
```

```
endif
pass
end-policy
```

Persistent Remarks

The following example shows how comments are placed in the policy to clarify the meaning of the entries in the set and the statements in the policy. The remarks are persistent, meaning they remain attached to the policy. For example, remarks are displayed in the output of the **show running-config** command. Adding remarks to the policy makes the policy easier to understand, modify at a later date, and troubleshoot if an unexpected behavior occurs.

```
prefix-set rfc1918
# These are the networks defined as private in RFC1918 (including
# all subnets thereof)
10.0.0.0/8 ge 8,
172.16.0.0/12 ge 12,
192.168.0.0/16 ge 16
end-set

route-policy quickstart-remarks
# Handle routes to RFC1918 networks
if destination in rfc1918 then
# Set the community such that we do not export the route
set community (no-export) additive

endif
end-policy
```

Routing Policy Configuration Basics

Route policies comprise series of statements and expressions that are bracketed with the **route-policy** and **end-policy** keywords. Rather than a collection of individual commands (one for each line), the statements within a route policy have context relative to each other. Thus, instead of each line being an individual command, each policy or set is an independent configuration object that can be used, entered, and manipulated as a unit.

Each line of a policy configuration is a logical subunit. At least one new line must follow the **then**, **else**, and **end-policy** keywords. A new line must also follow the closing parenthesis of a parameter list and the name string in a reference to an AS path set, community set, extended community set, or prefix set. At least one new line must precede the definition of a route policy, AS path set, community set, extended community set, or prefix set. One or more new lines can follow an action statement. One or more new lines can follow a comma separator in a named AS path set, community set, extended community set, or prefix set. A new line must appear at the end of a logical unit of policy expression and may not appear anywhere else.

Policy Definitions

Policy definitions create named sequences of policy statements. A policy definition consists of the CLI **route-policy** keyword followed by a name, a sequence of policy statements, and the **end-policy** keyword. For example, the following policy drops any route it encounters:

```
route-policy drop-everything
drop
```

```
end-policy
```

The name serves as a handle for binding the policy to protocols. To remove a policy definition, issue the **no route-policy name** command.

Policies may also refer to other policies such that common blocks of policy can be reused. This reference to other policies is accomplished by using the **apply** statement, as shown in the following example:

```
route-policy check-as-1234
if as-path passes-through '1234.5' then
  apply drop-everything
else
  pass
endif
end-policy
```

The **apply** statement indicates that the policy `drop-everything` should be executed if the route under consideration passed through autonomous system 1234.5 before it is received. If a route that has autonomous system 1234.5 in its AS path is received, the route is dropped; otherwise, the route is accepted without modification. This policy is an example of a hierarchical policy. Thus, the semantics of the **apply** statement are just as if the applied policy were cut and pasted into the applying policy:

```
route-policy check-as-1234-prime
  if as-path passes-through '1234.5' then
    drop
  else
    pass
  endif
end-policy
```

You may have as many levels of hierarchy as desired. However, many levels may be difficult to maintain and understand.

Parameterization

In addition to supporting reuse of policies using the **apply** statement, policies can be defined that allow for parameterization of some of the attributes. The following example shows how to define a parameterized policy named `param-example`. In this case, the policy takes one parameter, `$mytag`. Parameters always begin with a dollar sign and consist otherwise of any alphanumeric characters. Parameters can be substituted into any attribute that takes a parameter.

In the following example, a 16-bit community tag is used as a parameter:

```
route-policy param-example ($mytag)
  set community (1234:$mytag) additive
end-policy
```

This parameterized policy can then be reused with different parameterization, as shown in the following example. In this manner, policies that share a common structure but use different values in some of their

individual statements can be modularized. For details on which attributes can be parameterized, see the individual attribute sections.

```
route-policy origin-10
if as-path originates-from '10.5' then
apply param-example(10.5)
else
pass
endif
end-policy

route-policy origin-20
if as-path originates-from '20.5' then
apply param-example(20.5)
else
pass
endif
end-policy
```

The parameterized policy `param-example` provides a policy definition that is expanded with the values provided as the parameters in the `apply` statement. Note that the policy hierarchy is always maintained. Thus, if the definition of `param-example` changes, then the behavior of `origin_10` and `origin_20` changes to match.

The effect of the `origin-10` policy is that it adds the community `1234:10` to all routes that pass through this policy and have an AS path indicating the route originated from autonomous system 10. The `origin-20` policy is similar except that it adds to community `1234:20` for routes originating from autonomous system 20.

Parameterization at Attach Points

In addition to supporting parameterization using the `apply` statement described in the [Parameterization, on page 506](#), policies can also be defined that allow for parameterization the attributes at attach points. Parameterization is supported at all attach points.

In the following example, we define a parameterized policy "param-example". In this example, the policy takes two parameters "\$mymed" and "\$prefixset". Parameters always begin with a dollar sign, and consist otherwise of any alphanumeric characters. Parameters can be substituted into any attribute that takes a parameter. In this example we are passing a MED value and prefix set name as parameters.

```
route-policy param-example ($mymed, $prefixset)
if destination in $prefixset then
set med $mymed
endif
end-policy
```

This parameterized policy can then be reused with different parameterizations as shown in the example below. In this manner, policies that share a common structure but use different values in some of their individual statements can be modularized. For details on which attributes can be parameterized, see the individual attributes for each protocol.

```
router bgp 2
neighbor 10.1.1.1
remote-as 3
address-family ipv4 unicast
route-policy param-example(10, prefix_set1)
route-policy param-example(20, prefix_set2)
```

The parameterized policy `param-example` provides a policy definition that is expanded with the values provided as the parameters in the `neighbor route-policy in and out` statement.

Global Parameterization

RPL supports the definition of systemwide global parameters that can be used inside policy definition. Global parameters can be configured as follows:

```
Policy-global
  glbpathtype 'ebgp'
  glbtag '100'
end-global
```

The global parameter values can be used directly inside a policy definition similar to the local parameters of parameterized policy. In the following example, the *globalparam* argument, which makes use of the global parameters `glbpathtype` and `glbtag`, is defined for a nonparameterized policy.

```
route-policy globalparam
  if path-type is $glbpathtype then
    set tag $glbtag
  endif
end-policy
```

When a parameterized policy has a parameter name “collision” with a global parameter name, parameters local to policy definition take precedence, effectively masking off global parameters. In addition, a validation mechanism is in place to prevent the deletion of a particular global parameter if it is referred by any policy.

Semantics of Policy Application

This section discusses how routing policies are evaluated and applied. The following concepts are discussed:

Boolean Operator Precedence

Boolean expressions are evaluated in order of operator precedence, from left to right. The highest precedence operator is NOT, followed by AND, and then OR. The following expression:

```
med eq 10 and not destination in (10.1.3.0/24) or community matches-any ([10..25]:35)
```

if fully parenthesized to display the order of evaluation, would look like this:

```
(med eq 10 and (not destination in (10.1.3.0/24))) or community matches-any ([10..25]:35)
```

The inner NOT applies only to the destination test; the AND combines the result of the NOT expression with the Multi Exit Discriminator (MED) test; and the OR combines that result with the community test. If the order of operations are rearranged:

```
not med eq 10 and destination in (10.1.3.0/24) or community matches-any ([10..25]:35)
```

then the expression, fully parenthesized, would look like the following:

```
((not med eq 10) and destination in (10.1.3.0/24)) or community matches-any ([10..25]:35)
```

Multiple Modifications of the Same Attribute

When a policy replaces the value of an attribute multiple times, the last assignment wins because all actions are executed. Because the MED attribute in BGP is one unique value, the last value to which it gets set to wins. Therefore, the following policy results in a route with a MED value of 12:

```
set med 9
set med 10
set med 11
set med 12
```

This example is trivial, but the feature is not. It is possible to write a policy that effectively changes the value for an attribute. For example:

```
set med 8
if community matches-any cs1 then
set local-preference 122
if community matches-any cs2 then
set med 12
endif
endif
```

The result is a route with a MED of 8, unless the community list of the route matches both cs1 and cs2, in which case the result is a route with a MED of 12.

In the case in which the attribute being modified can contain only one value, it is easy to think of this case as the last statement wins. However, a few attributes can contain multiple values and the result of multiple actions on the attribute is cumulative rather than as a replacement. The first of these cases is the use of the **additive** keyword on community and extended community evaluation. Consider a policy of the form:

```
route-policy community-add
set community (10:23)
set community (10:24) additive
set community (10:25) additive
end-policy
```

This policy sets the community string on the route to contain all three community values: 10:23, 10:24, and 10:25.

The second of these cases is AS path prepending. Consider a policy of the form:

```
route-policy prepend-example
prepend as-path 2.5 3
prepend as-path 666.5 2
end-policy
```

This policy prepends 666.5 666.5 2.5 2.5 2.5 to the AS path. This prepending is a result of all actions being taken and to the AS path being an attribute that contains an array of values rather than a simple scalar value.

When Attributes Are Modified

A policy does not modify route attribute values until all tests have been completed. In other words, comparison operators always run on the initial data in the route. Intermediate modifications of the route attributes do not have a cascading effect on the evaluation of the policy. Take the following example:

```
ifmed eq 12 then
  set med 42
  if med eq 42 then
    drop
  endif
endif
```

This policy never executes the drop statement because the second test (med eq 42) sees the original, unmodified value of the MED in the route. Because the MED has to be 12 to get to the second test, the second test always returns false.

Default Drop Disposition

All route policies have a default action to drop the route under evaluation unless the route has been modified by a policy action or explicitly passed. Applied (nested) policies implement this disposition as though the applied policy were pasted into the point where it is applied.

Consider a policy to allow all routes in the 10 network and set their local preference to 200 while dropping all other routes. You might write the policy as follows:

```
route-policy two
  if destination in (10.0.0.0/8 ge 8 le 32) then
    set local-preference 200
  endif
end-policy

route-policy one
  apply two
end-policy
```

It may appear that policy one drops all routes because it neither contains an explicit **pass** statement nor modifies a route attribute. However, the applied policy does set an attribute for some routes and this disposition is passed along to policy one. The result is that policy one passes routes with destinations in network 10, and drops all others.

Control Flow

Policy statements are processed sequentially in the order in which they appear in the configuration. Policies that hierarchically reference other policy blocks are processed as if the referenced policy blocks had been directly substituted inline. For example, if the following policies are defined:

```
route-policy one
set weight 100
end-policy

route-policy two
set med 200
end-policy

route-policy three
apply two
set community (2:666) additive
end-policy

route-policy four
apply one
apply three
pass
end-policy
```

Policy four could be rewritten in an equivalent way as follows:

```
route-policy four-equivalent
set weight 100
set med 200
set community (2:666) additive
pass
end-policy
```



Note The **pass** statement is not required and can be removed to represent the equivalent policy in another way.

Policy Verification

Several different types of verification occur when policies are being defined and used.

Range Checking

As policies are being defined, some simple verifications, such as range checking of values, is done. For example, the MED that is being set is checked to verify that it is in a proper range for the MED attribute. However, this range checking cannot cover parameter specifications because they may not have defined values yet. These parameter specifications are verified when a policy is attached to an attach point. The policy repository also verifies that there are no recursive definitions of policy, and that parameter numbers are correct. At attach time, all policies must be well formed. All sets and policies that they reference must be defined and have valid values. Likewise, any parameter values must also be in the proper ranges.

Incomplete Policy and Set References

As long as a given policy is not attached at an attach point, the policy is allowed to refer to nonexistent sets and policies, which allows for freedom of workflow. You can build configurations that reference sets or policy blocks that are not yet defined, and then can later fill in those undefined policies and sets, thereby achieving much greater flexibility in policy definition. Every piece of policy you want to reference while defining a policy need not exist in the configuration. Thus, a user can define a policy sample that references the policy

bar using an **apply** statement even if the policy bar does not exist. Similarly, a user can enter a policy statement that refers to a nonexistent set.

However, the existence of all referenced policies and sets is enforced when a policy is attached. If you attempt to attach the policy sample with the reference to an undefined policy bar at an inbound BGP policy using the **neighbor 1.2.3.4 address-family ipv4 unicast policy sample in** command, the configuration attempt is rejected because the policy bar does not exist.

Likewise, you cannot remove a route policy or set that is currently in use at an attach point because this removal would result in an undefined reference. An attempt to remove a route policy or set that is currently in use results in an error message to the user.

A condition exists that is referred to as a null policy in which the policy bar exists but has no statements, actions, or dispositions in it. In other words, the policy bar does exist as follows:

```
route-policy bar
end-policy
```

This is a valid policy block. It effectively forces all routes to be dropped because it is a policy block that never modifies a route, nor does it include the pass statement. Thus, the default action of drop for the policy block is followed.

Attached Policy Modification

Policies that are in use do, on occasion, need to be modified. Traditionally, configuration changes are done by completely removing the relevant configuration and then re-entering it. However, this allows for a window of time in which no policy is attached and the default action takes place. RPL provides a mechanism for an atomic change so that if a policy is redeclared, or edited using a text editor, the new configuration is applied immediately—which allows for policies that are in use to be changed without having a window of time in which no policy is applied at the given attach point.

Verification of Attribute Comparisons and Actions

The policy repository knows which attributes, actions, and comparisons are valid at each attach point. When a policy is attached, these actions and comparisons are verified against the capabilities of that particular attach point. Take, for example, the following policy definition:

```
route-policy bad
set med 100
set level level-1-2
set ospf-metric 200
end-policy
```

This policy attempts to perform actions to set the BGP attribute med, IS-IS attribute level, and OSPF attribute cost. The system allows you to define such a policy, but it does not allow you to attach such a policy. If you had defined the policy bad and then attempted to attach it as an inbound BGP policy using the BGP configuration statement **neighbor 1.2.3.4 address-family ipv4 unicast route-policy bad in** the system would reject this configuration attempt. This rejection results from the verification process checking the policy and realizing that while BGP could set the MED, it has no way of setting the level or cost as the level and cost are attributes of IS-IS and OSPF, respectively. Instead of silently omitting the actions that cannot be done, the system generates an error to the user. Likewise, a valid policy in use at an attach point cannot be modified

in such a way as to introduce an attempt to modify a nonexistent attribute or to compare against a nonexistent attribute. The verifiers test for nonexistent attributes and reject such a configuration attempt.

Policy Statements

Four types of policy statements exist: remark, disposition (drop and pass), action (set), and if (comparator).

Remark

A remark is text attached to policy configuration but otherwise ignored by the policy language parser. Remarks are useful for documenting parts of a policy. The syntax for a remark is text that has each line prepended with a pound sign (#):

```
# This is a simple one-line remark.

# This
# is a remark
# comprising multiple
# lines.
```

In general, remarks are used between complete statements or elements of a set. Remarks are not supported in the middle of statements or within an inline set definition.

Unlike traditional !-comments in the CLI, RPL remarks persist through reboots and when configurations are saved to disk or a TFTP server and then loaded back onto the router.

Disposition

If a policy modifies a route, by default the policy accepts the route. RPL provides a statement to force the opposite—the **drop** statement. If a policy matches a route and executes a drop, the policy does not accept the route. If a policy does not modify the route, by default the route is dropped. To prevent the route from being dropped, the **pass** statement is used.

The **drop** statement indicates that the action to take is to discard the route. When a route is dropped, no further execution of policy occurs. For example, if after executing the first two statements of a policy the **drop** statement is encountered, the policy stops and the route is discarded.



Note All policies have a default **drop** action at the end of execution.

The **pass** statement allows a policy to continue executing even though the route has not been modified. When a policy has finished executing, any route that has been modified in the policy or any route that has received a pass disposition in the policy, successfully passes the policy and completes the execution. If route policy B_rp is applied within route policy A_rp, execution continues from policy A_rp to policy B_rp and back to policy A_rp provided prefix is not dropped by policy B_rp.

```
route-policy A_rp
  set community (10:10)
  apply B_rp
end-policy
!
```

```

route-policy B_rp
  if destination in (121.23.0.0/16 le 32, 155.12.0.0/16 le 32) then
    set community (121:155) additive
  endif
end-policy
!
```

By default, a route is **dropped** at the end of policy processing unless either the policy **modifies** a route attribute or it passes the route by means of an explicit **pass** statement. For example, if route-policy B is applied within route-policy A, then execution continues from policy A to policy B and back to policy A, provided the prefix is not dropped by policy B.

```

route-policy A
  if as-path neighbor-is '123' then
    apply B
    policy statement N
  endif
end-policy
```

Whereas the following policies pass all routes that they evaluate.

```

route-policy PASS-ALL
pass
end-policy

route-policy SET-LPREF
set local-preference 200
end-policy
```

In addition to being implicitly dropped, a route may be dropped by an **explicit drop** statement. **Drop** statements cause a route to be dropped immediately so that no further policy processing is done. Note also that a **drop** statement overrides any previously processed **pass** statements or attribute modifications. For example, the following policy drops all routes. The first **pass** statement is executed, but is then immediately overridden by the **drop** statement. The second **pass** statement never gets executed.

```

route-policy DROP-EXAMPLE
pass
drop
pass
end-policy
```

When one policy applies another, it is as if the applied policy were copied into the right place in the applying policy, and then the same drop-and-pass semantics are put into effect. For example, policies ONE and TWO are equivalent to policy ONE-PRIME:

```

route-policy ONE
  apply two
  if as-path neighbor-is '123' then
    pass
  endif
end-policy

route-policy TWO
```



```
if destination in (10.0.0.0/16 le 32) then
drop
endif
end-policy

route-policy ONE-PRIME
if destination in (10.0.0.0/16 le 32) then
drop
endif
if as-path neighbor-is '123' then
pass
endif
end-policy
```

Because the effect of an **explicit drop** statement is immediate, routes in 10.0.0.0/16 le 32 are dropped without any further policy processing. Other routes are then considered to see if they were advertised by autonomous system 123. If they were advertised, they are passed; otherwise, they are implicitly dropped at the end of all policy processing.

The **done** statement indicates that the action to take is to stop executing the policy and accept the route. When encountering a **done** statement, the route is passed and no further policy statements are executed. All modifications made to the route prior to the **done** statement are still valid.

Action

An action is a sequence of primitive operations that modify a route. Most actions, but not all, are distinguished by the **set** keyword. In a route policy, actions can be grouped together. For example, the following is a route policy comprising three actions:

```
route-policy actions
set med 217
set community (12:34) additive
delete community in (12:56)
end-policy
```

If

In its simplest form, an **if** statement uses a conditional expression to decide which actions or dispositions should be taken for the given route. For example:

```
if as-path in as-path-set-1 then
drop
endif
```

The example indicates that any routes whose AS path is in the set as-path-set-1 are dropped. The contents of the **then** clause may be an arbitrary sequence of policy statements.

The following example contains two action statements:

```
if origin is igp then
set med 42
prepend as-path 73.5 5
endif
```

The CLI provides support for the **exit** command as an alternative to the **endif** command.

The **if** statement also permits an **else** clause, which is executed if the if condition is false:

```
if med eq 8 then
set community (12:34) additive
else
set community (12:56) additive
endif
```

The policy language also provides syntax, using the **elseif** keyword, to string together a sequence of tests:

```
if med eq 150 then
set local-preference 10
elseif med eq 200 then
set local-preference 60
elseif med eq 250 then
set local-preference 110
else
set local-preference 0
endif
```

The statements within an **if** statement may themselves be **if** statements, as shown in the following example:

```
if community matches-any (12:34,56:78) then
if med eq 150 then
drop
endif
set local-preference 100
endif
```

This policy example sets the value of the local preference attribute to 100 on any route that has a community value of 12:34 or 56:78 associated with it. However, if any of these routes has a MED value of 150, then these routes with either the community value of 12:34 or 56:78 and a MED of 150 are dropped.



Note Policy grammar allows user to enter simple if statements with optional else clauses on the same line. However, the grammar is restricted to single action or disposition statement. For detailed command options, enter match statement on a separate line.

Boolean Conditions

In the previous section describing the **if** statement, all of the examples use simple Boolean conditions that evaluate to either true or false. RPL also provides a way to build compound conditions from simple conditions by means of Boolean operators.

Three Boolean operators exist: negation (**not**), conjunction (**and**), and disjunction (**or**). In the policy language, negation has the highest precedence, followed by conjunction, and then by disjunction. Parentheses may be used to group compound conditions to override precedence or to improve readability.

The following simple condition:

```
med eq 42
```

is true only if the value of the MED in the route is 42, otherwise it is false.

A simple condition may also be negated using the **not** operator:

```
not next-hop in (10.0.2.2)
```

Any Boolean condition enclosed in parentheses is itself a Boolean condition:

```
(destination in prefix-list-1)
```

A compound condition takes either of two forms. It can be a simple expression followed by the **and** operator, itself followed by a simple condition:

```
med eq 42 and next-hop in (10.0.2.2)
```

A compound condition may also be a simpler expression followed by the **or** operator and then another simple condition:

```
origin is igp or origin is incomplete
```

An entire compound condition may be enclosed in parentheses:

```
(med eq 42 and next-hop in (10.0.2.2))
```

The parentheses may serve to make the grouping of subconditions more readable, or they may force the evaluation of a subcondition as a unit.

In the following example, the highest-precedence **not** operator applies only to the destination test, the **and** operator combines the result of the **not** expression with the community test, and the **or** operator combines that result with the MED test.

```
med eq 10 or not destination in (10.1.3.0/24) and community matches-any ([12..34]:[56..78])
```

With a set of parentheses to express the precedence, the result is the following:

```
med eq 10 or ((not destination in (10.1.3.0/24)) and community matches-any ([12..34]:[56..78]))
```

The following is another example of a complex expression:

```
(origin is igp or origin is incomplete or not med eq 42) and next-hop in (10.0.2.2)
```

The left conjunction is a compound condition enclosed in parentheses. The first simple condition of the inner compound condition tests the value of the origin attribute; if it is Interior Gateway Protocol (IGP), then the inner compound condition is true. Otherwise, the evaluation moves on to test the value of the origin attribute again, and if it is incomplete, then the inner compound condition is true. Otherwise, the evaluation moves to check the next component condition, which is a negation of a simple condition.

apply

As discussed in the sections on policy definitions and parameterization of policies, the **apply** command executes another policy (either parameterized or unparameterized) from within another policy, which allows for the reuse of common blocks of policy. When combined with the ability to parameterize common blocks of policy, the **apply** command becomes a powerful tool for reducing repetitive configuration.

Attach Points

Policies do not become useful until they are applied to routes, and for policies to be applied to routes they need to be made known to routing protocols. In BGP, for example, there are several situations where policies can be used, the most common of these is defining import and export policy. The policy attach point is the point in which an association is formed between a specific protocol entity, in this case a BGP neighbor, and a specific named policy. It is important to note that a verification step happens at this point. Each time a policy is attached, the given policy and any policies it may apply are checked to ensure that the policy can be validly used at that attach point. For example, if a user defines a policy that sets the IS-IS level attribute and then attempts to attach this policy as an inbound BGP policy, the attempt would be rejected because BGP routes do not carry IS-IS attributes. Likewise, when policies are modified that are in use, the attempt to modify the policy is verified against all current uses of the policy to ensure that the modification is compatible with the current uses.

Each protocol has a distinct definition of the set of attributes (commands) that compose a route. For example, BGP routes may have a community attribute, which is undefined in OSPF. Routes in IS-IS have a level attribute, which is unknown to BGP. Routes carried internally in the RIB may have a tag attribute.

When a policy is attached to a protocol, the protocol checks the policy to ensure the policy operates using route attributes known to the protocol. If the protocol uses unknown attributes, then the protocol rejects the attachment. For example, OSPF rejects attachment of a policy that tests the values of BGP communities.

The situation is made more complex by the fact that each protocol has access to at least two distinct route types. In addition to native protocol routes, for example BGP or IS-IS, some protocol policy attach points operate on RIB routes, which is the common central representation. Using BGP as an example, the protocol provides an attach point to apply policy to routes redistributed from the RIB to BGP. An attach point dealing with two different kinds of routes permits a mix of operations: RIB attribute operations for matching and BGP attribute operations for setting.



Note The protocol configuration rejects attempts to attach policies that perform unsupported operations.

The following sections describe the protocol attach points, including information on the attributes (commands) and operations that are valid for each attach point.

See *Routing Command Reference for Cisco CRS Routers* for more information on the attributes and operations.

New para for test

BGP Policy Attach Points

This section describes each of the BGP policy attach points and provides a summary of the BGP attributes and operators.

Additional-Path

The additional-path attach point provides increased control based on various attribute match operations. This attach point is used to decide whether a route-policy should be used to select additional-paths for a BGP speaker to be able to send multiple paths for the prefix.

The add path enables BGP prefix independent convergence (PIC) at the edge routers.

This example shows how to set a route-policy "add-path-policy" to be used for enabling selection of additional paths:

```
router bgp 100
  address-family ipv4 unicast
  additional-paths selection route-policy add-path-policy
```

Dampening

The dampening attach point controls the default route-dampening behavior within BGP. Unless overridden by a more specific policy on the associate peer, all routes in BGP apply the associated policy to set their dampening attributes.

The following policy sets dampening values for BGP IPv4 unicast routes. Those routes that are more specific than a /25 take longer to recover after they have been dampened than routes that are less specific than /25.

```
route-policy sample_damp
  if destination in (0.0.0.0/0 ge 25) then
    set dampening halflife 30 others default
  else
    set dampening halflife 20 others default
  endif
end-policy

router bgp 2
  address-family ipv4 unicast
  bgp dampening route-policy sample_damp
  .
  .
  .
```

Default Originate

The default originate attach point allows the default route (0.0.0.0/0) to be conditionally generated and advertised to a peer, based on the presence of other routes. It accomplishes this configuration by evaluating the associated policy against routes in the Routing Information Base (RIB). If any routes pass the policy, the default route is generated and sent to the relevant peer.

The following policy generates and sends a default-route to the BGP neighbor 10.0.0.1 if any routes that match 10.0.0.0/8 ge 8 le 32 are present in the RIB.

```
route-policy sample-originate
  if rib-has-route in (10.0.0.0/8 ge 8 le 32) then
    pass
  endif
end-policy

router bgp 2
  neighbor 10.0.0.1
  remote-as 3
  address-family ipv4 unicast
  default-originate route-policy sample-originate
  .
  .
  .
```

Neighbor Export

The neighbor export attach point selects the BGP routes to send to a given peer or group of peers. The routes are selected by running the set of possible BGP routes through the associated policy. Any routes that pass the policy are then sent as updates to the peer or group of peers. The routes that are sent may have had their BGP attributes altered by the policy that has been applied.

The following policy sends all BGP routes to neighbor 10.0.0.5. Routes that are tagged with any community in the range 2:100 to 2:200 are sent with a MED of 100 and a community of 2:666. The rest of the routes are sent with a MED of 200 and a community of 2:200.

```
route-policy sample-export
  if community matches-any (2:[100-200]) then
    set med 100
    set community (2:666)
  else
    set med 200
    set community (2:200)
  endif
end-policy

router bgp 2
  neighbor 10.0.0.5
  remote-as 3
  address-family ipv4 unicast
  route-policy sample-export out
  .
  .
  .
```

Neighbor Import

The neighbor import attach point controls the reception of routes from a specific peer. All routes that are received by a peer are run through the attached policy. Any routes that pass the attached policy are passed to the BGP Routing Information Base (BRIB) as possible candidates for selection as best path routes.

When a BGP import policy is modified, it is necessary to rerun all the routes that have been received from that peer against the new policy. The modified policy may now discard routes that were previously allowed through, allow through previously discarded routes, or change the way the routes are modified. A new configuration option in BGP (**bgp auto-policy-soft-reset**) that allows this modification to happen automatically

in cases for which either soft reconfiguration is configured or the BGP route-refresh capability has been negotiated.

The following example shows how to receive routes from neighbor 10.0.0.1. Any routes received with the community 3:100 have their local preference set to 100 and their community tag set to 2:666. All other routes received from this peer have their local preference set to 200 and their community tag set to 2:200.

```
route-policy sample_import
  if community matches-any (3:100) then
    set local-preference 100
    set community (2:666)
  else
    set local-preference 200
    set community (2:200)
  endif
end-policy

router bgp 2
  neighbor 10.0.0.1
  remote-as 3
  address-family ipv4 unicast
    route-policy sample_import in
  .
  .
  .
```

Network

The network attach point controls the injection of routes from the RIB into BGP. A route policy attached at this point is able to set any of the valid BGP attributes on the routes that are being injected.

The following example shows a route policy attached at the network attach point that sets the well-known community no-export for any routes more specific than /24:

```
route-policy NetworkControl
  if destination in (0.0.0.0/0 ge 25) then
    set community (no-export) additive
  endif
end-policy

router bgp 2
  address-family ipv4 unicast
    network 172.16.0.5/27 route-policy NetworkControl
```

Redistribute

The redistribute attach point within OSPF injects routes from other routing protocol sources into the OSPF link-state database, which is done by selecting the routes it wants to import from each protocol. It then sets the OSPF parameters of cost and metric type. The policy can control how the routes are injected into OSPF by using the **set metric-type** or **set ospf-metric** command.

The following example shows how to redistribute routes from IS-IS instance instance_10 into OSPF instance 1 using the policy OSPF-redist. The policy sets the metric type to type-2 for all redistributed routes. IS-IS routes with a tag of 10 have their cost set to 100, and IS-IS routes with a tag of 20 have their OSPF cost set to 200. Any IS-IS routes not carrying a tag of either 10 or 20 are not be redistributed into the OSPF link-state database.

```

route-policy OSPF-redirect
  set metric-type type-2
  if tag eq 10 then
    set ospf cost 100
  elseif tag eq 20 then
    set ospf cost 200
  else
    drop
  endif
end-policy
router ospf 1
  redistribute isis instance_10 policy OSPF-redirect
  .
  .
  .

```

Show BGP

The `show bgp attach point` allows the user to display selected BGP routes that pass the given policy. Any routes that are not dropped by the attached policy are displayed in a manner similar to the output of the `show bgp` command.

In the following example, the `show bgp route-policy` command is used to display any BGP routes carrying a MED of 5:

```

route-policy sample-display
  if med eq 5 then
    pass
  endif
end-policy
!
show bgp route-policy sample-display

```

A `show bgp policy route-policy` command also exists, which runs all routes in the RIB past the named policy as if the RIB were an outbound BGP policy. This command then displays what each route looked like before it was modified and after it was modified, as shown in the following example:

show rpl route-policy test2

```

route-policy test2
  if (destination in (10.0.0.0/8 ge 8 le 32)) then
    set med 333
  endif
end-policy
!

```

show bgp

```

BGP router identifier 10.0.0.1, local AS number 2
BGP main routing table version 11
BGP scan interval 60 secs
Status codes:s suppressed, d damped, h history, * valid, > best
                i - internal, S stale
Origin codes:i - IGP, e - EGP, ? - incomplete
  Network          Next Hop          Metric LocPrf Weight Path
*> 10.0.0.0        10.0.1.2          10             0 3 ?
*> 10.0.0.0/9     10.0.1.2          10             0 3 ?

```



```

*> 10.0.0.0/10      10.0.1.2      10      0 3 ?
*> 10.0.0.0/11      10.0.1.2      10      0 3 ?
*> 10.1.0.0/16      10.0.1.2      10      0 3 ?
*> 10.3.30.0/24     10.0.1.2      10      0 3 ?
*> 10.3.30.128/25   10.0.1.2      10      0 3 ?
*> 10.128.0.0/9     10.0.1.2      10      0 3 ?
*> 10.255.0.0/24    10.0.101.2    1000    555    0 100 e
*> 10.255.64.0/24   10.0.101.2    1000    555    0 100 e
....

```

show bgp policy route-policy test2

```

10.0.0.0/8 is advertised to 10.0.101.2

Path info:
  neighbor:10.0.1.2      neighbor router id:10.0.1.2
  valid external best
Attributes after inbound policy was applied:
  next hop:10.0.1.2
  MET ORG AS
  origin:incomplete neighbor as:3 metric:10
  aspath:3
Attributes after outbound policy was applied:
  next hop:10.0.1.2
  MET ORG AS
  origin:incomplete neighbor as:3 metric:333
  aspath:2 3
...

```

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. See the *Implementing Routing Policy on Cisco IOS XR Software* module in the *Routing Configuration Guide for Cisco CRS Routers* for details on table policy use. See the *Cisco Express Forwarding Commands on Cisco IOS XR Software* module in the *IP Addresses and Services Command Reference for Cisco CRS Routers* for details on BGP policy accounting.

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.

Import

The import attach point provides control over the import of routes from the global VPN IPv4 table to a particular VPN routing and forwarding (VRF) instance.

For Layer 3 VPN networks, provider edge (PE) routers learn of VPN IPv4 routes through the Multiprotocol Internal Border Gateway Protocol (MP-iBGP) from other PE routers and automatically filters out route announcements that do not contain route targets that match any import route targets of its VRFs.

This automatic route filtering happens without RPL configuration; however, to provide more control over the import of routes in a VRF, you can configure a VRF import policy.

The following example shows how to perform matches based on a route target extended community and then sets the next hop. If the route has route target value 10:91, then the next hop is set to 172.16.0.1. If the route has route target value 11:92, then the next hop is set to 172.16.0.2. If the route has Site-of-Origin (SoO) value 10:111111 or 10:111222, then the route is dropped. All other non-matching routes are dropped.

```
route-policy bgpvrf_import
  if extcommunity rt matches-any (10:91) then
    set next-hop 172.16.0.1
  elseif extcommunity rt matches-every (11:92) then
    set next-hop 172.16.0.2
  elseif extcommunity soo matches-any (10:111111, 10:111222) then
    pass
  endif
end-policy

vrf vrf_import
  address-family ipv4 unicast
    import route-policy bgpvrf_import
  .
  .
  .
```



Note 'Set' is a valid operator for the 'med' attribute at the bgp import attach point.

Export

The export attach point provides control over the export of routes from a particular VRF to a global VPN IPv4 table.

For Layer 3 VPN networks, export route targets are added to the VPN IPv4 routes when VRF IPv4 routes are converted into VPN IPv4 routes and advertised through the MP-iBGP to other PE routers (or flow from one VRF to another within a PE router).

A set of export route targets is configured with the VRF without RPL configuration; however, to set route targets conditionally, you can configure a VRF export policy.

The following example shows some match and set operations supported for the export route policy. If a route matches 172.16.1.0/24 then the route target extended community is set to 10:101, and the weight is set to 211. If the route does not match 172.16.1.0/24 but the origin of the route is egp, then the local preference is set to 212 and the route target extended community is set to 10:101. If the route does not match those specified criteria, then the route target extended community 10:111222 is added to the route. In addition, RT 10:111222 is added to the route that matches any of the previous conditions as well.

```
route-policy bgpvrf_export
  if destination in (172.16.1.0/24) then
    set extcommunity rt (10:101)
    set weight 211
  elseif origin is egp then
    set local-preference 212
    set extcommunity rt (10:101)
  endif
  set extcommunity rt (10:111222) additive
end-policy

vrf vrf-export
```

```

address-family ipv4 unicast
  export route-policy bgpvrfl-export
  .
  .
  .

```



Note 'Set' is a valid operator for the 'med' attribute at the bgp export attach point.

Allocate-Label

The allocate-label attach point provides increased control based on various attribute match operations. This attach point is typically used in inter-AS option C to decide whether the label should be allocated or not when sending updates to the neighbor for the IPv4 labeled unicast address family. The attribute setting actions supported are for pass and drop.

The following example shows how to configure a route policy that passes the prefix 0.0.0.0 with prefix length 0. Label allocation happens only if prefix 0.0.0.0 exists.

```

route-policy label_policy
  if destination in (0.0.0.0/0) then
    pass
  endif
end-policy

router bgp 2
  vrf vrf1
    rd auto
  address-family ipv4 unicast
    allocate-label route-policy label-policy
  .
  .
  .

```

Retain Route-Target

The retain route target attach point within BGP allows the specification of match criteria based only on route target extended community. The attach point is useful at the route reflector (RR) or at the Autonomous System Boundary Router (ASBR).

Typically, an RR has to retain all IPv4 VPN routes to peer with its PE routers. These PEs might require routers tagged with different route target IPv4 VPN routes resulting in non-scalable RRs. You can achieve scalability if you configure an RR to retain routes with a defined set of route target extended communities, and a specific set of VPNs to service.

Another reason to use this attach point is for an ASBR. ASBRs do not require that VRFs be configured, but need this configuration to retain the IPv4 VPN prefix information.

The following example shows how to configure the route policy retainer and apply it to the retain route target attach point. The route is accepted if the route contains route target extended communities 10:615, 10:6150, and 15.15.15.15.15. All other non-matching routes are dropped.

```

extcommunity-set rt rtset1
  0:615,
  10:6150,

```

```

15.15.15.15.15:15
end-set

route-policy retainer
  if extcommunity rt matches-any rtset1 then
    pass
  endif
end-policy

router bgp 2
  address-family vpnv4 unicast
    retain route-target route-policy retainer
  .
  .
  .

```

Label-Mode

The label-mode attachpoint provides facility to choose label mode based on arbitrary match criteria such as prefix value, community. This attach point is typically used to set the type of label mode to per-ce or per-vrf or per-prefix based on deployment preferences. The attribute setting actions supported are for pass and drop.

This example shows label mode selection at VPNv4 AF (address family) level and at VRF IPv4 AF level:

```

route-policy set_label_mode
  set label-mode per-prefix
end-policy
!
router bgp 100
  address-family vpnv4 unicast
    vrf all
      label mode route-policy pass-all
    !
  !
  vrf abc
    rd 1:1
    address-family ipv4 unicast
      label mode route-policy set_label_mode
    !
  !
!
end

```

Neighbor-ORF

The neighbor-orf attach point provides the filtering of incoming BGP route updates using only prefix-based matching. In addition to using this as an inbound filter, the prefixes and disposition (drop or pass) are sent to upstream neighbors as an Outbound Route Filter (ORF) to allow them to perform filtering.

The following example shows how to configure a route policy orf-preset and apply it to the neighbor ORF attach point. The prefix of the route is dropped if it matches any prefix specified in orf-preset (172.16.1.0/24, 172.16.5.0/24, 172.16.11.0/24). In addition to this inbound filtering, BGP also sends these prefix entries to the upstream neighbor with a permit or deny so that the neighbor can filter updates before sending them on to their destination.

```

prefix-set orf-preset
  172.16.1.0/24,
  172.16.5.0/24,
  172.16.11.0/24
end-set

```

```

route-policy policy-orf
  if orf prefix in orf-preset then
    drop
  endif
  if orf prefix in (172.16.3.0/24, 172.16.7.0/24, 172.16.13.0/24) then
    pass
  endif

router bgp 2
  neighbor 1.1.1.1
  remote-as 3
  address-family ipv4 unicast
    orf route-policy policy-orf
  .
  .
  .

```

Next-hop

The next-hop attach point provides increased control based on protocol and prefix-based match operations. The attach point is typically used to decide whether to act on a next-hop notification (up or down) event.

Support for next-hop tracking allows BGP to monitor reachability for routes in the Routing Information Base (RIB) that can directly affect BGP prefixes. The route policy at the BGP next-hop attach point helps limit notifications delivered to BGP for specific prefixes. The route policy is applied on RIB routes. Typically, route policies are used in conjunction with next-hop tracking to monitor non-BGP routes.

The following example shows how to configure the BGP next-hop tracking feature using a route policy to monitor static or connected routes with the prefix 10.0.0.0 and prefix length 8.

```

route-policy nxthp_policy_A
  if destination in (10.0.0.0/8) and protocol in (static, connected) then
    pass
  endif
end-policy

router bgp 2
  address-family ipv4 unicast
    nexthop route-policy nxthp_policy_A
  .
  .
  .

```

Clear-Policy

The clear-policy attach point provides increased control based on various AS path match operations when using a **clear bgp** command. This attach point is typically used to decide whether to clear BGP flap statistics based on AS-path-based match operations.

The following example shows how to configure a route policy where the in operator evaluates to true if one or more of the regular expression matches in the set my-as-set successfully match the AS path associated with the route. If it is a match, then the **clear** command clears the associated flap statistics.

```

as-path-set my-as-set
  ios-regex '_12$',
  ios-regex '_13$'
end-set

```

```

route-policy policy_a
  if as-path in my-as-set then
    pass
  else
    drop
  endif
end-policy

clear bgp ipv4 unicast flap-statistics route-policy policy_a

```

Debug

The debug attach point provides increased control based on prefix-based match operations. This attach point is typically used to filter debug output for various BGP commands based on the prefix of the route.

The following example shows how to configure a route policy that will only pass the prefix 20.0.0.0 with prefix length 8; therefore, the debug output shows up only for that prefix.

```

route-policy policy_b
  if destination in (10.0.0.0/8) then
    pass
  else
    drop

  endif
end-policy

debug bgp update policy_b

```

BGP Attributes and Operators

This table summarizes the BGP attributes and operators per attach points.

Table 8: BGP Attributes and Operators

Attach Point	Attribute	Match	Set
additional-paths	path-selection	—	set
	community	matches-every is-empty matches-any	—

Attach Point	Attribute	Match	Set
aggregation	as-path	in is-local length neighbor-is originates-from passes-through unique-length	n/a
	as-path-length	is, ge, le, eq	n/a
	as-path-unique-length	is, ge, le, eq	n/a
	community	is-empty matches-any matches-every	set set additive delete in delete not in delete all
	destination	in	n/a
	extcommunity cost	n/a	set set additive
	local-preference	is, eg, ge, le	set
	med	is, eg, ge, le	set set+ set-
	next-hop	in	n/a
	origin	is	set
	source	in	n/a
	suppress-route	n/a	suppress-route
weight	n/a	set	

Attach Point	Attribute	Match	Set
allocate-label	as-path	in is-local length neighbor-is originates-from passes-through unique-length	n/a
	as-path-length	is, ge, le, eq	n/a
	as-path-unique-length	is, ge, le, eq	n/a
	community	is-empty matches-any matches-every	n/a
	destination	in	n/a
	label	n/a	set
	local-preference	is, ge, le, eq	n/a
	med	is, eg, ge, le	n/a
	next-hop	in	n/a
	origin	is	n/a
source	in	n/a	
clear-policy	as-path	in is-local length neighbor-is originates-from passes-through unique-length	n/a

Attach Point	Attribute	Match	Set
dampening	as-path	in is-local length neighbor-is originates-from passes-through unique-length	n/a
	as-path-length	is, ge, le, eq	n/a
	as-path-unique-length	is, ge, le, eq	n/a
	community	is-empty matches-any matches-every	n/a
	dampening	n/a	set dampening To set values that control the dampening (see Dampening , on page 519)
	destination	in	n/a
	local-preference	is, eg, ge, le	n/a
	med	is, eg, ge, le	n/a
	next-hop	in	n/a
	origin	is	n/a
	source	in	n/a
debug	destination	in	n/a

Attach Point	Attribute	Match	Set
default originate	as-path	n/a	prepend
	community community with `peeras`	n/a	set set additive
	extcommunity cost	n/a	set set additive
	extcommunity rt	n/a	set
	extcommunity soo	n/a	set
	local-preference	n/a	set
	med	n/a	set set + set -assign igp
	next-hop	n/a	set set-to-peer-address set-to-self
	origin	n/a	set
rib-has-route	in	n/a	

Attach Point	Attribute	Match	Set
export	as-path	in is-local length neighbor-is originates-from passes-through unique-length	n/a
	as-path-length	is, ge, le, eq	n/a
	as-path-unique-length	is, ge, le, eq	n/a
	community	is-empty matches-any matches-every	set set additive delete in delete not in delete all
	destination	in	n/a
	extcommunity rt	is-empty matches-any matches-every matches-within	set set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every matches-within	set set additive delete in delete not in delete all
	local-preference	is, eg, ge, le	set
	med	is, eg, ge, le	set
	next-hop	in	n/a
	origin	is	n/a
	source	in	n/a
weight	n/a	set	

Attach Point	Attribute	Match	Set
import	as-path	in is-local length neighbor-is originates-from passes-through unique-length	n/a
	as-path-length	is, ge, le, eq	n/a
	as-path-unique-length	is, ge, le, eq	n/a
	community	is-empty matches-any matches-every	n/a
	destination	in	n/a
	extcommunity rt	is-empty matches-any matches-every matches-within	n/a
	extcommunity soo	is-empty matches-any matches-every matches-within	n/a
	local-preference	is, ge, le, eq	set
	med	is, eg, ge, le	set
	next-hop	in	set set peer address set destination vrf
	origin	is	n/a
	source	in	n/a
weight	n/a	set	

Attach Point	Attribute	Match	Set
neighbor-in	as-path	in is-local length neighbor-is originates-from passes-through unique-length	prepend prepend most-recent replace
	as-path-length	is, ge, le, eq	n/a
	as-path-unique-length	is, ge, le, eq	n/a
	communitycommunity with 'peeras'	is-empty matches-any matches-every	set set additive delete in delete not in delete all
	destination	in	n/a
	extcommunity cost	n/a	set set additive
	extcommunity rt	is-empty matches-any matches-every matches-within	set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every matches-within	n/a
	local-preference	is, ge, le, eq	set
	med	is, eg, ge, le	set set+ set-
next-hop	in	set set peer address	

Attach Point	Attribute	Match	Set
	origin	is	set
	path-type	is	n/a
	source	in	n/a
	vpn-distinguisher	is	n/a
	weight	n/a	set

Attach Point	Attribute	Match	Set
neighbor-out	as-path	in is-local length neighbor-is originates-from passes-through unique-length	prepend prepend most-recent replace
	as-path-length	is, ge, le, eq	n/a
	as-path-unique-length	is, ge, le, eq	n/a
	communitycommunity with 'peeras'	is-empty matches-any matches-every	set set additive delete in delete not in delete all
	destination	in	n/a
	extcommunity cost	n/a	set set additive
	extcommunity rt	is-empty matches-any matches-every matches-within	set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every matches-within	n/a
	local-preference	is, eg, ge, le	set
	med	is, eg, ge, le	set set+ set-
next-hop	in	set set self	

Attach Point	Attribute	Match	Set
	origin	is	set
	path-type	is	n/a
	rd	in	n/a
	source	in	n/a
	unsuppress-route	n/a	unsuppress-route
	vpn-distinguisher	n/a	set
	weight	is	n/a
neighbor-orf	orf-prefix	in	n/a
network	as-path	n/a	prepend
	community	n/a	set set additive delete in delete not in delete all
	destination	in	n/a
	extcommunity cost	n/a	set set additive
	local-preference	n/a	set
	med	n/a	set set+ set-
	next-hop	n/a	set
	origin	n/a	set
	route-type	is	
	tag	is, eg, ge, le	n/a
	weight	n/a	set
next-hop	destination	in	n/a
	protocol	is,in	n/a

Attach Point	Attribute	Match	Set
redistribute	as-path	n/a	prepend
	community	n/a	set set additive delete in delete not in delete all
	destination	in	n/a
	extcommunity cost	n/a	set set additive
	local-preference	n/a	set
	med	n/a	set set+ set-
	next-hop	n/a	set
	origin	n/a	set
	rib-metric	is, eq, ge, le	n/a
	route-has-label	route-has-label	n/a
	route-type	is	n/a
	tag	is, eq, ge, le	n/a
	weight	n/a	set
retain-rt	extcommunity rt	is-empty matches-any matches-every matches-within	n/a

Attach Point	Attribute	Match	Set
show	as-path	in is-local length neighbor-is originates-from passes-through unique-length	n/a
	as-path-length	is, ge, le, eq	n/a
	as-path-unique-length	is, ge, le, eq	n/a
	community	is-empty matches-any matches-every	n/a
	destination	in	n/a
	extcommunity rt	is-empty matches-any matches-every matches-within	n/a
	extcommunity soo	is-empty matches-any matches-every matches-within	n/a
	med	is, eg, ge, le	n/a
	next-hop	in	n/a
	origin	is	n/a
source	in	n/a	

Attach Point	Attribute	Match	Set
table-policy	as-path	in is-local length neighbor-is originates-from passes-through unique-length	n/a
	as-path-length	is, ge, le, eq	n/a
	as-path-unique-length	is, ge, le, eq	n/a
	community	is-empty matches-any matches-every	n/a
	destination	in	n/a
	med	is, eg, ge, le	n/a
	next-hop	in	n/a
	origin	is	n/a
	rib-metric	n/a	set
	source	in	n/a
	tag	n/a	set
	traffic-index	n/a	set

Some BGP route attributes are inaccessible from some BGP attach points for various reasons. For example, the **set med igp-cost only** command makes sense when there is a configured `igp-cost` to provide a source value.

Default-Information Originate

The default-information originate attach point allows the user to conditionally inject the default route 0.0.0.0/0 into the OSPF link-state database, which is done by evaluating the attached policy. If any routes in the local RIB pass the policy, then the default route is inserted into the link-state database.

The following example shows how to generate a default route if any of the routes that match 10.0.0.0/8 ge 8 le 25 are present in the RIB:

```
route-policy ospf-originate
  if rib-has-route in (10.0.0.0/8 ge 8 le 25) then
    pass
  endif
```

```

end-policy

router ospf 1
  default-information originate policy ospf-originate
  .
  .
  .

```

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.

OSPF Policy Attach Points

This section describes each of the OSPF policy attach points and provides a summary of the OSPF attributes and operators.

Default-Information Originate

The default-information originate attach point allows the user to conditionally inject the default route 0.0.0.0/0 into the OSPF link-state database, which is done by evaluating the attached policy. If any routes in the local RIB pass the policy, then the default route is inserted into the link-state database.

The following example shows how to generate a default route if any of the routes that match 10.0.0.0/8 ge 8 le 25 are present in the RIB:

```
route-policy ospf-originate
  if rib-has-route in (10.0.0.0/8 ge 8 le 25) then
    pass
  endif
end-policy

router ospf 1
  default-information originate policy ospf-originate
  .
  .
  .
```

Redistribute

The redistribute attach point within OSPF injects routes from other routing protocol sources into the OSPF link-state database, which is done by selecting the routes it wants to import from each protocol. It then sets the OSPF parameters of cost and metric type. The policy can control how the routes are injected into OSPF by using the **set metric-type** or **set ospf-metric** command.

The following example shows how to redistribute routes from IS-IS instance instance_10 into OSPF instance 1 using the policy OSPF-redist. The policy sets the metric type to type-2 for all redistributed routes. IS-IS routes with a tag of 10 have their cost set to 100, and IS-IS routes with a tag of 20 have their OSPF cost set to 200. Any IS-IS routes not carrying a tag of either 10 or 20 are not be redistributed into the OSPF link-state database.

```
route-policy OSPF-redist
  set metric-type type-2
  if tag eq 10 then
    set ospf cost 100
  elseif tag eq 20 then
    set ospf cost 200
  else
    drop
  endif
end-policy
router ospf 1
  redistribute isis instance_10 policy OSPF-redist
  .
  .
  .
```

Area-in

The area-in attach point within OSPF allows you to filter inbound OSPF type-3 summary link-state advertisements (LSAs). The attach point provides prefix-based matching and hence increased control for filtering type-3 summary LSAs.

The following example shows how to configure the prefix for OSPF summary LSAs. If the prefix matches any of 111.105.3.0/24, 111.105.7.0/24, 111.105.13.0/24, it is accepted. If the prefix matches any of 111.106.3.0/24, 111.106.7.0/24, 111.106.13.0/24, it is dropped.

```

route-policy OSPF-area-in
  if destination in (
111.105.3.0/24,
111.105.7.0/24,
111.105.13.0/24) then
    drop
  endif
  if destination in (
111.106.3.0/24,
111.106.7.0/24,
111.106.13.0/24) then
    pass
  endif
end-policy

router ospf 1
  area 1
    route-policy OSPF-area-in in

```

Area-out

The area-out attach point within OSPF allows you to filter outbound OSPF type-3 summary LSAs. The attach point provides prefix-based matching and, hence, increased control for filtering type-3 summary LSAs.

The following example shows how to configure the prefix for OSPF summary LSAs. If the prefix matches any of 211.105.3.0/24, 211.105.7.0/24, 211.105.13.0/24, it is announced. If the prefix matches any of .105.3.0/24, 212.105.7.0/24, 212.105.13.0/24, it is dropped and not announced.

```

route-policy OSPF-area-out
  if destination in (
211.105.3.0/24,
211.105.7.0/24,
211.105.13.0/24) then
    drop
  endif
  if destination in (
212.105.3.0/24,
212.105.7.0/24,
212.105.13.0/24) then
    pass
  endif
end-policy

router ospf 1
  area 1
    route-policy OSPF-area-out out

```

SPF Prefix-priority

The spf-prefix-priority attach point within OSPF allows you to define the route policy to apply to OSPFv2 prefix prioritization.

OSPF Attributes and Operators

This table summarizes the OSPF attributes and operators per attach points.

Table 9: OSPF Attributes and Operators

Attach Point	Attribute	Match	Set
distribute-list-in-area	destination	in	n/a
	rib-metric	in	n/a
	tag	eq, ge, is, le	n/a
distribute-list-in-instance	destination	in	n/a
	rib-metric	in	n/a
	tag	eq, ge, is, le	n/a
distribute-list-in-interface	destination	in	n/a
	rib-metric	in	n/a
	tag	eq, ge, is, le	n/a
default-information originate	ospf-metric	n/a	set
	metric-type	n/a	set
	rib-has-route	in	n/a
	tag	n/a	set
redistribute	destination	in	n/a
	metric-type	n/a	set
	next-hop	in	n/a
	ospf-metric	n/a	set
	rib-metric	is, le, ge, eq	n/a
	route-has-level	route-has-level	n/a
	route-type	is	n/a
	tag	is, le, ge, le	set
area-in	destination	in	n/a
area-out	destination	in	n/a
spf-prefix-priority	destination	in	n/a
	spf-priority	n/a	set
	tag	is, le, ge, eq	n/a

Distribute-list in

The distribute-list in attach point within OSPF allows use of route policies to filter OSPF prefixes. The distribute-list in route-policy can be configured at OSPF instance, area, and interface levels. The route-policy used in the distribute-list in command supports match statements, "destination" and "rib-metric". The "set" commands are not supported in the route-policy.

These are examples of valid route-policies for "distribute-list in":

```
route-policy DEST
  if destination in (10.10.10.10/32) then
    drop
  else
    pass
  endif
end-policy

route-policy METRIC
  if rib-metric ge 10 and rib-metric le 19 then
    drop
  else
    pass
  endif
end-policy

prefix-set R-PFX
  10.10.10.30
end-set

route-policy R-SET
  if destination in R-PFX and rib-metric le 20 then
    pass
  else
    drop
  endif
end-policy
```

OSPFv3 Policy Attach Points

This section describes each of the OSPFv3 policy attach points and provides a summary of the OSPFv3 attributes and operators.

Default-Information Originate

The default-information originate attach point allows the user to conditionally inject the default route 0::/0 into the OSPFv3 link-state database, which is done by evaluating the attached policy. If any routes in the local RIB pass the policy, then the default route is inserted into the link-state database.

The following example shows how to generate a default route if any of the routes that match 2001::/96 are present in the RIB:

```
route-policy ospfv3-originate
  if rib-has-route in (2001::/96) then
    pass
  endif
end-policy
```



```
router ospfv3 1
  default-information originate policy ospfv3-originate
  .
  .
```

Redistribute

The redistribute attach point within OSPFv3 injects routes from other routing protocol sources into the OSPFv3 link-state database, which is done by selecting the route types it wants to import from each protocol. It then sets the OSPFv3 parameters of cost and metric type. The policy can control how the routes are injected into OSPFv3 by using the **metric type** command.

The following example shows how to redistribute routes from BGP instance 15 into OSPF instance 1 using the policy OSPFv3-redirect. The policy sets the metric type to type-2 for all redistributed routes. BGP routes with a tag of 10 have their cost set to 100, and BGP routes with a tag of 20 have their OSPFv3 cost set to 200. Any BGP routes not carrying a tag of either 10 or 20 are not be redistributed into the OSPFv3 link-state database.

```
route-policy OSPFv3-redirect
  set metric-type type-2
  if tag eq 10 then
    set extcommunity cost 100
  elseif tag eq 20 then
    set extcommunity cost 200
  else
    drop
  endif
end-policy

router ospfv3 1
  redistribute bgp 15 policy OSPFv3-redirect
  .
  .
```

OSPFv3 Attributes and Operators

This table summarizes the OSPFv3 attributes and operators per attach points.

Table 10: OSPFv3 Attributes and Operators

Attach Point	Attribute	Match	Set
default-information originate	ospf-metric	n/a	set
	metric-type	n/a	set
	rib-has-route	in	n/a
	tag	n/a	set

Attach Point	Attribute	Match	Set
redistribute	destination	in	n/a
	metric-type	n/a	set
	next-hop	in	n/a
	ospf-metric	n/a	set
	rib-metric	is, le, ge, eq	n/a
	route-has-level	route-has-level	n/a
	route-type	is	n/a
	tag	is, le, ge, eq	set

IS-IS Policy Attach Points

This section describes each of the IS-IS policy attach points and provides a summary of the IS-IS attributes and operators.

Redistribute

The redistribute attach point within IS-IS allows routes from other protocols to be readvertised by IS-IS. The policy is a set of control structures for selecting the types of routes that a user wants to redistribute into IS-IS. The policy can also control which IS-IS level the routes are injected into and at what metric values.

The following describes an example. Here, routes from IS-IS instance 1 are redistributed into IS-IS instance instance_10 using the policy ISIS-redist. This policy sets the level to level-1-2 for all redistributed routes. IS-IS routes with a tag of 10 have their metric set to 100, and IS-IS routes with a tag of 20 have their IS-IS metric set to 200. Any IS-IS routes not carrying a tag of either 10 or 20 are not be redistributed into the IS-IS database.

```

route-policy ISIS-redist
  set level level-1-2
  if tag eq 10 then
    set isis-metric 100
  elseif tag eq 20 then
    set isis-metric 200
  else
    drop
  endif
end-policy

router isis instance_10
  address-family ipv4 unicast
    redistribute isis 1 policy ISIS-redist
  .
  .

```

Default-Information Originate

The default-information originate attach point within IS-IS allows the default route 0.0.0.0/0 to be conditionally injected into the IS-IS route database.

The following example shows how to generate an IPv4 unicast default route if any of the routes that match 10.0.0.0/8 ge 8 le 25 is present in the RIB. The cost of the IS-IS route is set to 100 and the level is set to level-1-2 on the default route that is injected into the IS-IS database.

```
route-policy isis-originate
  if rib-has-route in (10.0.0.0/8 ge 8 le 25) then
    set metric 100
    set level level-1-2
  endif
end-policy

router isis instance_10
  address-family ipv4 unicast
    default-information originate policy isis_originate
  .
```

Inter-area-propagate

The inter-area-propagate attach point within IS-IS allows the prefixes to be conditionally propagated from one level to another level within the same IS-IS instance.

The following example shows how to allow prefixes to be leaked from the level 1 LSP into the level 2 LSP if any of the prefixes match 10.0.0.0/8 ge 8 le 25.

```
route-policy isis-propagate
  if destination in (10.0.0.0/8 ge 8 le 25) then
    pass
  endif
end-policy

router isis instance_10
  address-family ipv4 unicast
    propagate level 1 into level 2 policy isis-propagate
  .
```

IS-IS Attributes and Operators

This table summarizes the IS-IS attributes and operators per attach points.

Table 11: IS-IS Attributes and Operators

Attach Point	Attribute	Match	Set
redistribution	tag	is, le, ge	n/a
	route-type	is Note The following route-type cannot be matched: <i>ospf-nssa-type-1</i> and <i>ospf-nssa-type-2</i>	n/a
	destination	in	n/a
	next-hop	in	n/a
	mpls-label	route-has-label	n/a
	rib-metric	is, le, ge, eq	n/a
	level	n/a	set
	isis-metric	n/a	set
	metric	n/a	set
	metric-type	n/a	set
default-information originate	rib-has-route	in	n/a
	level	n/a	set
	isis-metric	n/a	set
	tag	n/a	set
inter-area-propagate	destination	in	n/a

EIGRP Policy Attach Points

This section describes each of the EIGRP policy attach points and provides a summary of the EIGRP attributes and operators.

Default-Accept-In

The default-accept-in attach point allows you to set and reset the conditional default flag for EIGRP routes by evaluating the attached policy.

The following example shows a policy that sets the conditional default flag for all routes that match 10.0.0.0/8 and longer prefixes up to 10.0.0.0/25:

```
route-policy eigrp-cd-policy-in
  if destination in (10.0.0.0/8 ge 8 le 25) then
```

```

        pass
      endif
    end-policy
  !
  router eigrp 100
    address-family ipv4
      default-information allowed in route-policy eigrp-cd-policy-in
    .
    .
    .

```

Default-Accept-Out

The default-accept-out attach point allows you to set and reset the conditional default flag for EIGRP routes by evaluating the attached policy.

The following example shows a policy that sets the conditional default flag for all routes that match 100.10.0.0/16:

```

    route-policy eigrp-cd-policy-out
      if destination in (
200.10.0.0/16) then
        pass
      endif
    end-policy
  !
  router eigrp 100
    address-family ipv4
      default-information allowed out route-policy eigrp-cd-policy-out
    .
    .
    .

```

Policy-In

The policy-in attach point allows you to filter and modify inbound EIGRP routes. This policy is applied to all interfaces for which there is no interface inbound route policy.

The following example shows the command under EIGRP:

```

  router eigrp 100
    address-family ipv4
      route-policy global-policy-in in
    .
    .
    .

```

Policy-Out

The policy-out attach point allows you to filter and modify outbound EIGRP routes. This policy is applied to all interfaces for which there is no interface outbound route policy.

The following example shows the command under EIGRP:

```

  router eigrp 100
    address-family ipv4
      route-policy global-policy-out out
    .

```

```

.
.

```

If-Policy-In

The if-policy-in attach point allows you to filter routes received on a particular EIGRP interface. The following example shows an inbound policy for GigabitEthernet interface 0/2/0/3:

```

router eigrp 100
  address-family ipv4
    interface GigabitEthernet0/2/0/3
      route-policy if-filter-policy-in in
    .
  .

```

If-Policy-Out

The if-policy-out attach point allows you to filter routes sent out on a particular EIGRP interface. The following example shows an outbound policy for GigabitEthernet interface 0/2/0/3:

```

router eigrp 100
  address-family ipv4
    interface GigabitEthernet0/2/0/3
      route-policy if-filter-policy-out out
    .
  .

```

Redistribute

The redistribute attach point in EIGRP allows you to filter redistributed routes from other routing protocols and modify some routing parameters before installing the route in the EIGRP database. The following example shows a policy filter redistribution of RIP routes into EIGRP.

```

router-policy redistribute-rip
  if destination in (100.1.1.0/24) then
    set eigrp-metric 5000000 4000 150 30 2000
  else
    set tag 200
  endif
end-policy

router eigrp 100
  address-family ipv4
    redistribute rip route-policy redistribute-rip
  .
  .

```

EIGRP Attributes and Operators

This table summarizes the EIGRP attributes and operators per attach points.

Table 12: EIGRP Attributes and Operators

Attach Point	Attribute	Match	Set
default-accept-in	destination	in	n/a
default-accept-out	destination	in	n/a
if-policy-in	destination	in	n/a
	next-hop	in	n/a
	eigrp-metric	n/a	add, set
	tag	is, eq, ge, le	set
if-policy-out	destination	in	n/a
	next-hop	in	n/a
	protocol	is, in	n/a
	eigrp-metric	n/a	add, set
	tag	is, eq, ge, le	set
policy-in	destination	in	n/a
	next-hop	in	n/a
	eigrp-metric	n/a	add, set
	tag	is, eq, ge, le	set
policy-out	destination	in	n/a
	next-hop	in	n/a
	protocol	is, in	n/a
	eigrp-metric	n/a	add, set
	tag	is, eq, ge, le	set

Attach Point	Attribute	Match	Set
redistribute	destination	in	n/a
	next-hop	in	n/a
	route-has-level	route-has-level	n/a
	eigrp-metric	n/a	add, set
	rib-metric	is, le, ge, eq	n/a
	route-type	is	n/a
	tag	is, le, ge, eq	set

RIP Policy Attach Points

This section describes each of the RIP policy attach points and provides a summary of the RIP attributes and operators.

Default-Information Originate

The default-information originate attach point allows you to conditionally inject the default route 0.0.0.0/0 into RIP updates by evaluating the attached policy. If any routes in the local RIB pass the policy, then the default route is inserted.

The following example shows how to generate a default route if any of the routes that match 10.0.0.0/8 ge 8 le 25 are present in the RIB:

```
route-policy rip-originate
  if rib-has-route in (10.0.0.0/8 ge 8 le 25) then
    pass
  endif
end-policy

router rip
  default-information originate route-policy rip-originate
```

Redistribute

The redistribution attach point within RIP allows you to inject routes from other routing protocol sources into the RIP database.

The following example shows how to inject OSPF routes into RIP:

```
route-policy redistrib-ospf
  set rip-metric 5
end-policy

router rip
  redistribute ospf 1 route-policy redistrib-ospf
```


Global-Inbound

The global-inbound attach point for RIP allows you to filter or update inbound RIP routes that match a route policy.

The following example shows how to filter the inbound RIP routes that match the route policy named rip-in:

```
router rip
  route-policy rip-in in
```

Global-Outbound

The global-outbound attach point for RIP allows you to filter or update outbound RIP routes that match a route-policy.

The following example shows how to filter the outbound RIP routes that match the route policy named rip-out:

```
router rip
  route-policy rip-out out
```

Interface-Inbound

The interface-inbound attach point allows you to filter or update inbound RIP routes that match a route policy for a specific interface.

The following example shows how to filter inbound RIP routes that match the route policy for interface 0/1/0/1:

```
router rip
  interface GigabitEthernet0/1/0/1
    route-policy rip-in in
```

Interface-Outbound

The interface-outbound attach point allows you to filter or update outbound RIP routes that match a route policy for a specific interface.

The following example shows how to filter outbound RIP routes that match the route policy for interface 0/2/0/1:

```
router rip
  interface GigabitEthernet0/2/0/1
    route-policy rip-out out
```

RIP Attributes and Operators

This table summarizes the RIP attributes and operators per attach points.

Table 13: RIP Attributes and Operators

Attach Point	Attribute	Match	Set
default-information originate	next-hop	n/a	set
	rip-metric	n/a	set
	rip-tag	n/a	set
	rib-has-route	in	n/a
global-inbound	destination	in	n/a
	next-hop	in	n/a
	rip-metric	n/a	add
global-outbound	destination	in	n/a
	next-hop	in	n/a
	protocol	is, in	n/a
	rip-metric	n/a	add
interface-inbound	destination	in	n/a
	next-hop	in	n/a
	rip-metric	n/a	add
interface-outbound	destination	in	n/a
	next-hop	in	n/a
	protocol	is, in	n/a
	rip-metric	n/a	add
redistribute	destination	in	n/a

Attach Point	Attribute	Match	Set
	tag	is, eq, ge, le	n/a
	route-type	is	n/a
	next-hop	in	set
	route-has-level	route-has-level	n/a
	rib-metric	is, eq, ge, le	n/a
	rip-metric	n/a	set
	rip-tag	n/a	set

PIM Policy Attach Points

This section describes the PIM policy **rpf-topology** attach point and provides a summary of the PIM attributes and operators.

rpf-topology

The rpf-topology attach point is to set the Reverse Path Forwarding (RPF) to any default or non-default tables for particular Sources and/or Groups.

For example, the following policy sets the rpf-topology to table t201 if the destination address is either 225.0.0.1 or 225.0.0.11 then table t201 will be used to figure out the reverse path, else if the destination is 225.0.0.3 or 225.0.0.13 default table will be used to figure out the reverse forwarding path.

```

route-policy mt4-p1
  if destination in (225.0.0.1, 225.0.0.11) then
    set rpf-topology ipv4 multicast topology t201
  elseif destination in (225.0.0.3, 225.0.0.13) then
    pass
  endif
end-policy
.
.
.

router pim
  rpf topology route-policy mt4-p1

```

To set an extranet VRF for RPF lookups, use the **rpf topology route-policy route-policy name** command in PIM. For example:

```

router pim
  rpf topology route-policy extranet-vrf-policy

route-policy extranet-vrf-policy
  if destination in (225.1.1.1/32) then
    set rpf-topology vrf parent-vrf
  else
    pass

```

```
end-policy
```



Note Set the `rpf-topology` attribute for PIM to default VRF to extranet the IGMP joins for Multicast VRF Override to function.

PIM Attributes and Operators

This table summarizes the PIM attributes and operators for the **rpf-topology** attach point.

Table 14: PIM Attributes and Operators

Attach Point	Attribute	Match	Set
rpf-topology	destination	in	ra
	source	in	ra
	rpf-topology	n/a	st

Nondestructive Editing of Routing Policy

The Nondestructive Editing of Routing Policy changes the default exit behavior under routing policy configuration mode to abort the configuration.

The default **exit** command acts as end-policy, end-set, or end-if. If the **exit** command is executed under route policy configuration mode, the changes are applied and configuration is updated. This destructs the existing policy. The **rpl set-exit-as-abort** command allows to overwrite the default behavior of the **exit** command under the route policy configuration mode.

Attached Policy Modification

Policies that are in use do, on occasion, need to be modified. In the traditional configuration model, a policy modification would be done by completely removing the policy and re-entering it. However, this model allows for a window of time in which no policy is attached and default actions to be used, which is an opportunity for inconsistencies to exist. To close this window of opportunity, you can modify a policy in use at an attach point by respecifying it, which allows for policies that are in use to be changed, without having a window of time in which no policy is applied at the given attach point.



Note A route policy or set that is in use at an attach point cannot be removed because this removal would result in an undefined reference. An attempt to remove a route policy or set that is in use at an attach point results in an error message to the user.

Nonattached Policy Modification

As long as a given policy is not attached at an attach point, the policy is allowed to refer to nonexistent sets and policies. Configurations can be built that reference sets or policy blocks that are not yet defined, and then later those undefined policies and sets can be filled in. This method of building configurations gives much greater flexibility in policy definition. Every piece of policy you want to reference while defining a policy need not exist in the configuration. Thus, you can define a policy sample1 that references a policy sample2 using an apply statement even if the policy sample2 does not exist. Similarly, you can enter a policy statement that refers to a nonexistent set.

However, the existence of all referenced policies and sets is enforced when a policy is attached. Thus, if a user attempts to attach the policy sample1 with the reference to an undefined policy sample2 at an inbound BGP policy using the statement **neighbor 1.2.3.4 address-family ipv4 unicast policy sample1 in**, the configuration attempt is rejected because the policy sample2 does not exist.

Editing Routing Policy Configuration Elements

RPL is based on statements rather than on lines. That is, within the begin-end pair that brackets policy statements from the CLI, a new line is merely a separator, the same as a space character.

The CLI provides the means to enter and delete route policy statements. RPL provides a means to edit the contents of the policy between the begin-end brackets, using a text editor. The following text editors are available on Cisco IOS XR software for editing RPL policies:

- Nano (default)
- Emacs
- Vim

Editing Routing Policy Configuration Elements Using the Nano Editor

To edit the contents of a routing policy using the Nano editor, use the following CLI command in EXEC mode:

```
edit route-policy  
  
    name  
  
nano
```

A copy of the route policy is copied to a temporary file and the editor is launched. After editing, enter Ctrl-X to save the file and exit the editor. The available editor commands are displayed on screen.

Detailed information on using the Nano editor is available at this URL: <http://www.nano-editor.org/>.

Not all Nano editor features are supported on Cisco IOS XR software.

Editing Routing Policy Configuration Elements Using the Emacs Editor

To edit the contents of a routing policy using the Emacs editor, use the following CLI command in EXEC mode:

```

edit

route-policy

name

emacs

```

A copy of the route policy is copied to a temporary file and the editor is launched. After editing, save the editor buffer by using the Ctrl-X and Ctrl-S keystrokes. To save and exit the editor, use the Ctrl-X and Ctrl-C keystrokes. When you quit the editor, the buffer is committed. If there are no parse errors, the configuration is committed:

```

RP/0/RP0/CPU0:router# edit route-policy policy_A
-----
== MicroEMACS 3.8b () == rpl_edit.139281 ==
  if destination in (2001::/8) then
    drop
  endif
end-policy
!

== MicroEMACS 3.8b () == rpl_edit.139281 ==
Parsing.
83 bytes parsed in 1 sec (82)bytes/sec
Committing.
1 items committed in 1 sec (0)items/sec
Updating.
Updated Commit database in 1 sec

```

If there are parse errors, you are asked whether editing should continue:

```

RP/0/RP0/CPU0:router#edit route-policy policy_B
== MicroEMACS 3.8b () == rpl_edit.141738
route-policy policy_B
  set metric-type type_1
  if destination in (2001::/8) then
    drop
  endif
end-policy
!
== MicroEMACS 3.8b () == rpl_edit.141738 ==
Parsing.
105 bytes parsed in 1 sec (103)bytes/sec

% Syntax/Authorization errors in one or more commands.!! CONFIGURATION
FAILED DUE TO SYNTAX/AUTHORIZATION ERRORS

```

```
set metric-type type_1
if destination in (2001::/8) then
    drop
endif
end-policy
!
```

Continue editing? [no]:

If you answer **yes**, the editor continues on the text buffer from where you left off. If you answer **no**, the running configuration is not changed and the editing session is ended.

Editing Routing Policy Configuration Elements Using the Vim Editor

Editing elements of a routing policy with Vim (Vi IMproved) is similar to editing them with Emacs except for some feature differences such as the keystrokes to save and quit. To write to a current file and exit, use the **:wq** or **:x** or **ZZ** keystrokes. To quit and confirm, use the **:q** keystrokes. To quit and discard changes, use the **:q!** keystrokes.

You can reference detailed online documentation for Vim at this URL: <http://www.vim.org/>

Editing Routing Policy Configuration Elements Using CLI

The CLI allows you to enter and delete route policy statements. You can complete a policy configuration block by entering applicable commands such as **end-policy** or **end-set**. Alternatively, the CLI interpreter allows you to use the **exit** command to complete a policy configuration block. The **abort** command is used to discard the current policy configuration and return to global configuration mode.

Editing Routing Policy Language set elements Using XML

RPL supports editing set elements using XML. Entries can be appended, prepended, or deleted to an existing set without replacing it through XML.

Hierarchical Policy Conditions

The Hierarchical Policy Conditions feature enables the ability to specify a route policy within the "if" statement of another route policy. This ability enables route-policies to be applied for configurations that are based on hierarchical policies.

With the Hierarchical Policy Conditions feature, Cisco IOS XR RPL supports Apply Condition policies that can be used with various types of Boolean operators along with various other matching statements.

Apply Condition Policies

Apply Condition policies, which Cisco IOS XR RPL supports, allow usage of a route-policy within an "if" statement of another route-policy.

Consider route-policy configurations *Parent*, *Child A*, and *Child B*:

```
route-policy Child A
if destination in (10.10.0.0/16) then
    set local-pref 111
endif
end-policy
!
```

```

route-policy Child B
  if as-path originates-from '222' then
    set community (333:222) additive
  endif
end-policy
!

route-policy Parent
  if apply Child A and apply Child B then
    set community (333:333) additive
  else
    set community (333:444) additive
  endif
end-policy
!
```

In the above scenarios, whenever the policy *Parent* is executed, the decision of the "if" condition in that is selected based on the result of policies *Child A* and *Child B*. The policy *Parent* is equivalent to policy *merged* as given below:

```

route-policy merged
  if destination in (10.10.0.0/16) and as-path originates-from '222' then
    set local-pref 111
    set community (333:222, 333:333) additive
  elseif destination in (10.10.0.0/16) then /*Only Policy Child A is pass */
    set local-pref 111
    set community (333:444) additive /*From else block */
  elseif as-path originates-from '222' then /*Only Policy Child B is pass */
    set community (333:222, 333:444) additive /*From else block */
  else
    set community (333:444) additive /*From else block */
  endif
end-policy
```

Apply Conditions can be used with parameters and are supported on all attach points and on all clients. Hierarchical Apply Conditions can be used without any constraints on a cascaded level.

Existing route policy semantics can be expanded to include this Apply Condition:

```

Route-policy policy_name
  If apply policyA and apply policyB then
    Set med 100
  Else if not apply policyD then
    Set med 200
  Else
    Set med 300
  Endif
End-policy
```


Behavior of pass/drop/done RPL Statements for Simple Hierarchical Policies

This table describes the behavior of **pass/drop/done** RPL statements, with a possible sequence for executing the **done** statement for Simple Hierarchical Policies.

Route-policies with simple hierarchical policies	Possible done statement execution sequence	Behavior
pass	pass Continue_list	Marks the prefix as "acceptable" and continues with execution of continue_list statements.
drop	Stmts_list drop	Rejects the route immediately on hitting the drop statement and stops policy execution.
done	Stmts_list done	Accepts the route immediately on hitting the done statement and stops policy execution.
pass followed by done	pass Statement_list done	Exits immediately at the done statement with "accept route".
drop followed by done	drop Statement list done	This is an invalid scenario at execution point of time. Policy terminates execution at the drop statement itself, without going through the statement list or the done statement; the prefix will be rejected or dropped.

Behavior of pass/drop/done RPL Statements for Hierarchical Policy Conditions

This section describes the behavior of **pass/drop/done** RPL statements, with a possible sequence for executing the **done** statement for Hierarchical Policy Conditions.

Terminology for policy execution: "true-path", "false-path", and "continue-path".

```
Route-policy parent
  If apply hierarchical_policy_condition then
    TRUE-PATH      : if hierarchical_policy_condition returns TRUE then this path will
                    be executed.
  Else
    FALSE-PATH     : if hierarchical_policy_condition returns FALSE then this path will
                    be executed.
  End-if
  CONTINUE-PATH   : Irrespective of the TRUE/FALSE this path will be executed.
End-policy
```

Hierarchical policy conditions	Possible done statement execution sequence	Behavior
pass	pass Continue_list	Marks the return value as "true" and continues execution within the same policy condition. If there is no statement after " pass ", returns "true".
pass followed by done	pass or set action statement Stmt_list done	Marks the return value as "true" and continues execution till the done statement. Returns "true" to the apply policy condition to take "true-path".
done	Stmt_list without pass or set operation DONE	Returns " false". Condition takes "false-path".
drop	Stmt_list drop Stmt_list	The prefix is dropped or rejected.

Nested Wildcard Apply Policy

The hierarchical constructs of Routing Policy Language (RPL) allows one policy to refer to another policy. The referred or called policy is known as a child policy. The policy from which another policy is referred is called calling or parent policy. A calling or parent policy can nest multiple child policies for attachment to a common set of BGP neighbors. The nested wildcard apply policy allows wildcard (*) based apply nesting. The wildcard operation permits declaration of a generic apply statement that calls all policies that contain a specific defined set of alphanumeric characters, defined on the router.

A wildcard is specified by placing an asterisk (*) at the end of the policy name in an apply statement. Passing parameters to wildcard policy is not supported. The wildcard indicates that any value for that portion of the apply policy matches.

To illustrate nested wildcard apply policy, consider this policy hierarchy:

```
route-policy Nested_Wilcard
apply service_policy_customer*
end-policy

route-policy service_policy_customer_a
if destination in prfx_set_customer_a then
set extcommunity rt (1:1) additive
endif
end-policy

route-policy service_policy_customer_b
if destination in prfx_set_customer_b then
set extcommunity rt (1:1) additive
endif
end-policy
```

```

route-policy service_policy_customer_c
if destination in prfx_set_customer_c then
set extcommunity rt (1:1) additive
endif
end-policy

```

Here, a single parent apply statement (apply service_policy_customer*) calls (inherits) all child policies that contain the identified character string "service_policy_customer". As each child policy is defined globally, the parent dynamically nests the child policies based on the policy name. The parent is configured once and inherits each child policy on demand. There is no direct association between the parent and the child policies beyond the wildcard match statement.

Wildcards for Route Policy Sets

Route policies are defined in a modular form, and comprise of sets of comparative statements. Using wildcards to define a range of sets, significantly reduces the complexity of a policy.

Wildcards can be used to define a range of prefix sets, community sets, AS-path sets, or extended community sets. For information on using wildcards in policy sets, see [Use Wildcards For Routing Policy Sets, on page 565](#).

Use Wildcards For Routing Policy Sets

This section describes examples of configuring routing policy sets with wildcards.

Use Wildcards for Prefix Sets

Use the following example to configure a routing policy with wildcards for prefix sets.

1. Configure the required prefix sets in the global configuration mode.

```

RP/0/RP0/CPU0:router (config) # prefix-set pfx_set1
RP/0/RP0/CPU0:router (config-pfx) # 1.2.3.4/32
RP/0/RP0/CPU0:router (config-pfx) # end-set
RP/0/RP0/CPU0:router (config) # prefix-set pfx_set2
RP/0/RP0/CPU0:router (config-pfx) # 2.2.2.2/32
RP/0/RP0/CPU0:router (config-pfx) # end-set

```

2. Configure a route policy with wildcards to refer to the prefix sets.

```

RP/0/RP0/CPU0:router (config) # route-policy WILDCARD_PREFIX_SET
RP/0/RP0/CPU0:router (config-rpl) # if destination in prefix-set* then pass else drop endif
RP/0/RP0/CPU0:router (config-rpl) # end-policy

```

This route policy configuration accepts routes with the prefixes mentioned in the two prefix sets, and drops all other non-matching routes.

3. Commit your configuration.

```

RP/0/RP0/CPU0:router (config) # commit

```

This completes the configuration of routing policy with wildcards for prefix sets. For detailed information on prefix sets, see [prefix-set, on page 500](#).

Use Wildcards for AS-Path Sets

Use the following example to configure a routing policy with wildcards for AS-path sets.

1. Configure the required AS-path sets in the global configuration mode.

```
RP/0/RP0/CPU0:router(config)# as-path-set AS_SET1
RP/0/RP0/CPU0:router(config-as)# ios-regex '_22$',
RP/0/RP0/CPU0:router(config-as)# ios-regex '_25$'
RP/0/RP0/CPU0:router(config-as)# end-set
RP/0/RP0/CPU0:router(config)# as-path-set AS_SET2
RP/0/RP0/CPU0:router(config-as)# ios-regex '_42$',
RP/0/RP0/CPU0:router(config-as)# ios-regex '_47$'
RP/0/RP0/CPU0:router(config-as)# end-set
```

2. Configure a route policy with wildcards to refer to the AS-path sets.

```
RP/0/RP0/CPU0:router(config)# route-policy WILDCARD_AS_SET
RP/0/RP0/CPU0:router(config-rpl)# if as-path in as-path-set* then pass else drop endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

This route policy configuration accepts routes with AS-path attributes as mentioned in the two AS-path sets, and drops all other non-matching routes.

3. Commit your configuration.

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

This completes the configuration of routing policy with wildcards for AS-path sets. For detailed information on AS-path sets, see [as-path-set](#), on page 496.

Use Wildcards for Community Sets

Use the following example to configure a routing policy with wildcards for community sets.

1. Configure the required community sets in the global configuration mode.

```
RP/0/RP0/CPU0:router(config)# community-set CSET1
RP/0/RP0/CPU0:router(config-comm)# 12:24,
RP/0/RP0/CPU0:router(config-comm)# 12:36,
RP/0/RP0/CPU0:router(config-comm)# 12:72
RP/0/RP0/CPU0:router(config-comm)# end-set
RP/0/RP0/CPU0:router(config)# community-set CSET2
RP/0/RP0/CPU0:router(config-comm)# 24:12,
RP/0/RP0/CPU0:router(config-comm)# 24:42,
RP/0/RP0/CPU0:router(config-comm)# 24:64
RP/0/RP0/CPU0:router(config-comm)# end-set
```

2. Configure a route policy with wildcards to refer to the community sets.

```
RP/0/RP0/CPU0:router(config)# route-policy WILDCARD_COMMUNITY_SET
RP/0/RP0/CPU0:router(config-rpl)# if community matches-any community-set* then pass else
drop endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

This route policy configuration accepts routes with community set values as mentioned in the two community sets, and drops all other non-matching routes.

3. Commit your configuration.

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

This completes the configuration of routing policy with wildcards for community sets. For detailed information on community path sets, see [community-set, on page 496](#).

Use Wildcards for Extended Community Sets

Use the following example to configure a routing policy with wildcards for extended community sets.

1. Configure the extended community sets in the global configuration mode.

```
RP/0/RP0/CPU0:router (config) # extcommunity-set rt RT_SET1
RP/0/RP0/CPU0:router (config-ext) # 1.2.3.4:555,
RP/0/RP0/CPU0:router (config-ext) # 1234:555
RP/0/RP0/CPU0:router (config-ext) # end-set
RP/0/RP0/CPU0:router (config) # extcommunity-set rt RT_SET2
RP/0/RP0/CPU0:router (config-ext) # 1.1.1.1:777,
RP/0/RP0/CPU0:router (config-ext) # 1111:777
RP/0/RP0/CPU0:router (config-ext) # end-set
```

2. Configure a route policy with wildcards to refer to the extended community sets.

```
RP/0/RP0/CPU0:router (config) # route-policy WILDCARD_EXT_COMMUNITY_SET
RP/0/RP0/CPU0:router (config-rpl) # if extcommunity rt matches-any extcommunity-set* then
pass else drop endif
RP/0/RP0/CPU0:router (config-rpl) # end-policy
```

This route policy configuration accepts routes with extended community set values as mentioned in the two extended community sets, and drops all other non-matching routes.

3. Commit your configuration.

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

This completes the configuration of routing policy with wildcards for extended community sets. For detailed information on extended community path sets, see [extcommunity-set, on page 497](#).

Use Wildcards for Route Distinguisher Sets

Use the following example to configure a routing policy with wildcards for route distinguisher sets.

1. Configure the route distinguisher sets in the global configuration mode.

```
RP/0/RP0/CPU0:router (config) # rd-set rd_set_demo
RP/0/RP0/CPU0:router (config-rd) # 10.0.0.1/8:77,
RP/0/RP0/CPU0:router (config-rd) # 10.0.0.2:888,
RP/0/RP0/CPU0:router (config-rd) # 65000:777
RP/0/RP0/CPU0:router (config-rd) # end-set
RP/0/RP0/CPU0:router (config) # rd-set rd_set_demo2
RP/0/RP0/CPU0:router (config-rd) # 20.0.0.1/7:99,
RP/0/RP0/CPU0:router (config-rd) # 4784:199
RP/0/RP0/CPU0:router (config-rd) # end-set
```

2. Configure a route policy with wildcards to refer to the route distinguisher set.

```
RP/0/RP0/CPU0:router (config) # route-policy use_rd_set
RP/0/RP0/CPU0:router (config-rpl) # if rd in rd-set* then set local-preference 100
RP/0/RP0/CPU0:router (config-rpl-if) # elseif rd in(10.0.0.2:888, 10.0.0.2:999) then set
local-preference 300
```

```
RP/0/RP0/CPU0:router(config-rpl-elseif)# endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

3. Commit your configuration.

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

4. (Optional) Verify your configuration.

```
RP/0/RP0/CPU0:router(config)# show configuration
...
Building configuration...
!! IOS XR Configuration 0.0.0
!
rd-set rd_set_demo
  10.0.0.1/8:77,
  10.0.0.2:888,
  65000:777
end-set
!
!
rd-set rd_set_demo2
  20.0.0.1/7:99,
  4784:199
end-set
!

route-policy use_rd_set
  if rd in rd-set* then
    set local-preference 100
  elseif rd in (10.0.0.2:888, 10.0.0.2:999) then
    set local-preference 300
  endif
end-policy
!
end
```

This completes the configuration of routing policy with wildcards for route distinguisher sets. For more information on route distinguisher sets, see [rd-set](#), on page 502.

Use Wildcards for OSPF Area Sets

Use the following example to configure a routing policy with wildcards for OSPF area sets.

1. Configure the OSPF area set in the global configuration mode.

```
RP/0/RP0/CPU0:router(config)# ospf-area-set ospf_area_set_demo1
RP/0/RP0/CPU0:router(config-ospf-area)# 10.0.0.1,
RP/0/RP0/CPU0:router(config-ospf-area)# 3553
RP/0/RP0/CPU0:router(config-ospf-area)# end-set

RP/0/RP0/CPU0:router(config)# ospf-area-set ospf_area_set_demo2
RP/0/RP0/CPU0:router(config-ospf-area)# 20.0.0.2,
RP/0/RP0/CPU0:router(config-ospf-area)# 3673
RP/0/RP0/CPU0:router(config-ospf-area)# end-set
```

2. Configure a route policy with wildcards to refer to the OSPF area set.

```
RP/0/RP0/CPU0:router(config)# route-policy use_ospf_area_set
RP/0/RP0/CPU0:router(config-rpl)# if ospf-area in ospf-area-set* then set ospf-metric
200
```

```
RP/0/RP0/CPU0:router(config-rpl-if)# elseif ospf-area in( 10.0.0.1, 10.0.0.2 )then set
ospf-metric 300
RP/0/RP0/CPU0:router(config-rpl-elseif)# endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

3. Commit your configuration.

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

4. (Optional) Verify your configuration.

```
RP/0/RP0/CPU0:router(config)# show configuration
Building configuration...
!! IOS XR Configuration 0.0.0
!
ospf-area-set ospf_area_set_demo1
  10.0.0.1,
  3553
end-set
!
!
ospf-area-set ospf_area_set_demo2
  20.0.0.2,
  3673
end-set
!

route-policy use_ospf_area_set
  if ospf-area in ospf-area-set* then
    set ospf-metric 200
  elseif ospf-area in (10.0.0.1, 10.0.0.2) then
    set ospf-metric 300
  endif
end-policy
!
end
```

This completes the configuration of routing policy with wildcards for OSPF area sets.

VRF Import Policy Enhancement

The VRF RPL based import policy feature provides the ability to perform import operation based solely on import route-policy, by matching on route-targets (RTs) and other criteria specified within the policy. No need to explicitly configure import RTs under global VRF-address family configuration mode. If the import RTs and import route-policy is already defined, then the routes will be imported from RTs configured under import RT and then follows the route-policy attached at import route-policy.

Use the **source rt import-policy** command under VRF sub-mode of VPN address-family configuration mode to enable this feature.

Flexible L3VPN Label Allocation Mode

The flexible L3VPN label allocation feature provides the ability to set label allocation mode using a route-policy, where different allocation modes can be used for different sets of prefixes. Thus, label mode can be chosen based on arbitrary match criteria such as prefix value and community.

Use the **label mode** command to set the MPLS/VPN label mode based on prefix value. The Label-Mode attach point enables you to choose label mode based on any arbitrary criteria.

Match Aggregated Route

The Match Aggregated Route feature helps to match BGP aggregated route from the non-aggregated route. BGP can aggregate a group of routes into a single prefix before sending updates to a neighbor. With Match Aggregated Route feature, route policy separates this aggregated route from other routes.

Remove Private AS in Inbound Policy

BGP appends its own as-path before sending out packets to neighbors. When a packet traverses multiple iBGP neighbors, the as-path structure will have many private autonomous systems (AS) in them. The Remove Private AS in Inbound Policy will give the capability to delete those private autonomous systems using RPL route-policy. The **remove as-path private-as** command removes autonomous systems (AS) with AS number 64512 through 65535.

Set Administrative Distance

The Set Administrative Distance modifies the administrative distance of each of the individual prefixes in BGP. When RIB has two routes to the same destination, RIB chooses the route with lower administrative distance for forwarding. The **set-administrative-distance** command sets the administrative distance of BGP route to a value such that RIB chooses the route which is required.

How to Implement Routing Policy

This section contains the following procedures:

Defining a Route Policy

This task explains how to define a route policy.



Note

- If you want to modify an existing routing policy using the command-line interface (CLI), you must redefine the policy by completing this task.
- Modifying the RPL scale configuration may take a long time.
- BGP may crash either due to large scale RPL configuration changes, or during consecutive RPL changes. To avoid BGP crash, wait until there are no messages in the BGP In/Out queue before committing further changes.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name* [*parameter1* , *parameter2* , . . . , *parameterN*]
3. **end-policy**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	route-policy <i>name</i> [<i>parameter1</i> , <i>parameter2</i> , ..., <i>parameterN</i>] Example: RP/0/RP0/CPU0:router(config)# route-policy sample1	Enters route-policy configuration mode. • After the route-policy has been entered, a group of commands can be entered to define the route-policy.
Step 3	end-policy Example: RP/0/RP0/CPU0:router(config-rpl)# end-policy	Ends the definition of a route policy and exits route-policy configuration mode.
Step 4	commit	

Attaching a Routing Policy to a BGP Neighbor

This task explains how to attach a routing policy to a BGP neighbor.

Before you begin

A routing policy must be preconfigured and well defined prior to it being applied at an attach point. If a policy is not predefined, an error message is generated stating that the policy is not defined.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 labeled-unicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **ipv6 labeled-unicast** | **vpn4 unicast** | **vpn6 unicast** }
5. **route-policy** *policy-name* { **in** | **out** }
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router bgp <i>as-number</i> Example: RP/0/RP0/CPU0:router(config)# router bgp 125	Configures a BGP routing process and enters router configuration mode. • The <i>as-number</i> argument identifies the autonomous system in which the router resides. Valid values are from 0 to 65535. Private autonomous system numbers that can be used in internal networks range from 64512 to 65535.

	Command or Action	Purpose
Step 3	neighbor <i>ip-address</i> Example: RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.20	Specifies a neighbor IP address.
Step 4	address-family { ipv4 unicast ipv4 multicast ipv4 labeled-unicast ipv4 tunnel ipv4 mdt ipv6 unicast ipv6 multicast ipv6 labeled-unicast vpnv4 unicast vpn6 unicast } Example: RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast	Specifies the address family.
Step 5	route-policy <i>policy-name</i> { in out } Example: RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy example1 in	Attaches the route-policy, which must be well formed and predefined.
Step 6	commit	

Modifying a Routing Policy Using a Text Editor

This task explains how to modify an existing routing policy using a text editor. See [Editing Routing Policy Configuration Elements, on page 559](#) for information on text editors.

SUMMARY STEPS

1. **edit** { **route-policy** | **prefix-set** | **as-path-set** | **community-set** | **extcommunity-set** { **rt** | **soo** } | **policy-global** | **rd-set** } *name* [**nano** | **emacs** | **vim** | **inline** { **add** | **prepend** | **remove** } *set-element*]
2. **show rpl route-policy** [*name* [**detail**] | **states** | **brief**]
3. **show rpl prefix-set** [*name* | **states** | **brief**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	edit { route-policy prefix-set as-path-set community-set extcommunity-set { rt soo } policy-global rd-set } <i>name</i> [nano emacs vim inline { add prepend remove } <i>set-element</i>] Example: RP/0/RP0/CPU0:router# edit route-policy sample1	Identifies the route policy, prefix set, AS path set, community set, or extended community set name to be modified. <ul style="list-style-type: none"> • A copy of the route policy, prefix set, AS path set, community set, or extended community set is copied to a temporary file and the editor is launched. • After editing with Nano, save the editor buffer and exit the editor by using the Ctrl-X keystroke.

	Command or Action	Purpose
		<ul style="list-style-type: none"> • After editing with Emacs, save the editor buffer by using the Ctrl-X and Ctrl-S keystrokes. To save and exit the editor, use the Ctrl-X and Ctrl-C keystrokes. • After editing with Vim, to write to a current file and exit, use the :wq or :x or ZZ keystrokes. To quit and confirm, use the :q keystrokes. To quit and discard changes, use the :q! keystrokes.
Step 2	show rpl route-policy [<i>name</i> [detail] states brief] Example: RP/0/RP0/CPU0:router# show rpl route-policy sample2	(Optional) Displays the configuration of a specific named route policy. <ul style="list-style-type: none"> • Use the detail keyword to display all policies and sets that a policy uses. • Use the states keyword to display all unused, inactive, and active states. • Use the brief keyword to list the names of all extended community sets without their configurations.
Step 3	show rpl prefix-set [<i>name</i> states brief] Example: RP/0/RP0/CPU0:router# show rpl prefix-set prefixset1	(Optional) Displays the contents of a named prefix set. <ul style="list-style-type: none"> • To display the contents of a named AS path set, community set, or extended community set, replace the prefix-set keyword with as-path-set, community-set, or extcommunity-set, respectively.

Configuration Examples for Implementing Routing Policy

This section provides the following configuration examples:

Routing Policy Definition: Example

In the following example, a BGP route policy named sample1 is defined using the **route-policy name** command. The policy compares the network layer reachability information (NLRI) to the elements in the prefix set test. If it evaluates to true, the policy performs the operations in the *then* clause. If it evaluates to false, the policy performs the operations in the *else* clause, that is, sets the MED value to 200 and adds the community 2:100 to the route. The final steps of the example commit the configuration to the router, exit configuration mode, and display the contents of route policy sample1.

```
configure
route-policy sample1
  if destination in test then
    drop
  else
    set med 200
    set community (2:100) additive
  endif
end-policy
```

```

end
show config running route-policy sample1
Building configuration...
route-policy sample1
  if destination in test then
    drop
  else
    set med 200
    set community (2:100) additive
  endif
end-policy

```

Simple Inbound Policy: Example

The following policy discards any route whose network layer reachability information (NLRI) specifies a prefix longer than /24, and any route whose NLRI specifies a destination in the address space reserved by RFC 1918. For all remaining routes, it sets the MED and local preference, and adds a community to the list in the route.

For routes whose community lists include any values in the range from 101:202 to 106:202 that have a 16-bit tag portion containing the value 202, the policy prepends autonomous system number 2 twice, and adds the community 2:666 to the list in the route. Of these routes, if the MED is either 666 or 225, then the policy sets the origin of the route to incomplete, and otherwise sets the origin to IGP.

For routes whose community lists do not include any of the values in the range from 101:202 to 106:202, the policy adds the community 2:999 to the list in the route.

```

prefix-set too-specific
  0.0.0.0/0 ge 25 le 32
end-set

prefix-set rfc1918
  10.0.0.0/8 le 32,
  172.16.0.0/12 le 32,
  192.168.0.0/16 le 32
end-set

route-policy inbound-tx
  if destination in too-specific or destination in rfc1918 then
    drop
  endif
  set med 1000
  set local-preference 90
  set community (2:1001) additive
  if community matches-any ([101..106]:202) then
    prepend as-path 2.30 2
    set community (2:666) additive
    if med is 666 or med is 225 then
      set origin incomplete
    else
      set origin igp
    endif
  else
    set community (2:999) additive
  endif
end-policy

router bgp 2
  neighbor 10.0.1.2 address-family ipv4 unicast route-policy inbound-tx in

```

Modular Inbound Policy: Example

The following policy example shows how to build two inbound policies, in-100 and in-101, for two different peers. In building the specific policies for those peers, the policy reuses some common blocks of policy that may be common to multiple peers. It builds a few basic building blocks, the policies common-inbound, filter-bogons, and set-lpref-prepend.

The filter-bogons building block is a simple policy that filters all undesirable routes, such as those from the RFC 1918 address space. The policy set-lpref-prepend is a utility policy that can set the local preference and prepend the AS path according to parameterized values that are passed in. The common-inbound policy uses these filter-bogons building blocks to build a common block of inbound policy. The common-inbound policy is used as a building block in the construction of in-100 and in-101 along with the set-lpref-prepend building block.

This is a simple example that illustrates the modular capabilities of the policy language.

```
prefix-set bogon
 10.0.0.0/8 ge 8 le 32,
 0.0.0.0,
 0.0.0.0/0 ge 27 le 32,
 192.168.0.0/16 ge 16 le 32
end-set
!
route-policy in-100
  apply common-inbound
  if community matches-any ([100..120]:135) then
    apply set-lpref-prepend (100,100,2)
    set community (2:1234) additive
  else
    set local-preference 110
  endif
  if community matches-any ([100..666]:[100..999]) then
    set med 444
    set local-preference 200
    set community (no-export) additive
  endif
end-policy
!
route-policy in-101
  apply common-inbound
  if community matches-any ([101..200]:201) then
    apply set-lpref-prepend(100,101,2)
    set community (2:1234) additive
  else
    set local-preference 125
  endif
end-policy
!
route-policy filter-bogons
  if destination in bogon then
  drop
  else
  pass
  endif
end-policy
!
route-policy common-inbound
  apply filter-bogons
  set origin igp
  set community (2:333)
end-policy
```

```

!
route-policy set-lpref-prepend($lpref,$as,$prependcnt)
  set local-preference $lpref
  prepend as-path $as $prependcnt
end-policy

```

Use Wildcards For Routing Policy Sets

This section describes examples of configuring routing policy sets with wildcards.

Use Wildcards for Prefix Sets

Use the following example to configure a routing policy with wildcards for prefix sets.

1. Configure the required prefix sets in the global configuration mode.

```

RP/0/RP0/CPU0:router(config)# prefix-set pfx_set1
RP/0/RP0/CPU0:router(config-pfx)# 1.2.3.4/32
RP/0/RP0/CPU0:router(config-pfx)# end-set
RP/0/RP0/CPU0:router(config)# prefix-set pfx_set2
RP/0/RP0/CPU0:router(config-pfx)# 2.2.2.2/32
RP/0/RP0/CPU0:router(config-pfx)# end-set

```

2. Configure a route policy with wildcards to refer to the prefix sets.

```

RP/0/RP0/CPU0:router(config)# route-policy WILDCARD_PREFIX_SET
RP/0/RP0/CPU0:router(config-rpl)# if destination in prefix-set* then pass else drop endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy

```

This route policy configuration accepts routes with the prefixes mentioned in the two prefix sets, and drops all other non-matching routes.

3. Commit your configuration.

```

RP/0/RP0/CPU0:router(config)# commit

```

This completes the configuration of routing policy with wildcards for prefix sets. For detailed information on prefix sets, see [prefix-set](#), on page 500.

Use Wildcards for AS-Path Sets

Use the following example to configure a routing policy with wildcards for AS-path sets.

1. Configure the required AS-path sets in the global configuration mode.

```

RP/0/RP0/CPU0:router(config)# as-path-set AS_SET1
RP/0/RP0/CPU0:router(config-as)# ios-regex '_22$',
RP/0/RP0/CPU0:router(config-as)# ios-regex '_25$'
RP/0/RP0/CPU0:router(config-as)# end-set
RP/0/RP0/CPU0:router(config)# as-path-set AS_SET2
RP/0/RP0/CPU0:router(config-as)# ios-regex '_42$',
RP/0/RP0/CPU0:router(config-as)# ios-regex '_47$'
RP/0/RP0/CPU0:router(config-as)# end-set

```

2. Configure a route policy with wildcards to refer to the AS-path sets.

```

RP/0/RP0/CPU0:router(config)# route-policy WILDCARD_AS_SET

```

```
RP/0/RP0/CPU0:router(config-rpl)# if as-path in as-path-set* then pass else drop endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

This route policy configuration accepts routes with AS-path attributes as mentioned in the two AS-path sets, and drops all other non-matching routes.

3. Commit your configuration.

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

This completes the configuration of routing policy with wildcards for AS-path sets. For detailed information on AS-path sets, see [as-path-set, on page 496](#).

Use Wildcards for Community Sets

Use the following example to configure a routing policy with wildcards for community sets.

1. Configure the required community sets in the global configuration mode.

```
RP/0/RP0/CPU0:router(config)# community-set CSET1
RP/0/RP0/CPU0:router(config-comm)# 12:24,
RP/0/RP0/CPU0:router(config-comm)# 12:36,
RP/0/RP0/CPU0:router(config-comm)# 12:72
RP/0/RP0/CPU0:router(config-comm)# end-set
RP/0/RP0/CPU0:router(config)# community-set CSET2
RP/0/RP0/CPU0:router(config-comm)# 24:12,
RP/0/RP0/CPU0:router(config-comm)# 24:42,
RP/0/RP0/CPU0:router(config-comm)# 24:64
RP/0/RP0/CPU0:router(config-comm)# end-set
```

2. Configure a route policy with wildcards to refer to the community sets.

```
RP/0/RP0/CPU0:router(config)# route-policy WILDCARD_COMMUNITY_SET
RP/0/RP0/CPU0:router(config-rpl)# if community matches-any community-set* then pass else
drop endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

This route policy configuration accepts routes with community set values as mentioned in the two community sets, and drops all other non-matching routes.

3. Commit your configuration.

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

This completes the configuration of routing policy with wildcards for community sets. For detailed information on community path sets, see [community-set, on page 496](#).

Use Wildcards for Extended Community Sets

Use the following example to configure a routing policy with wildcards for extended community sets.

1. Configure the extended community sets in the global configuration mode.

```
RP/0/RP0/CPU0:router(config)# extcommunity-set rt RT_SET1
RP/0/RP0/CPU0:router(config-ext)# 1.2.3.4:555,
RP/0/RP0/CPU0:router(config-ext)# 1234:555
RP/0/RP0/CPU0:router(config-ext)# end-set
RP/0/RP0/CPU0:router(config)# extcommunity-set rt RT_SET2
RP/0/RP0/CPU0:router(config-ext)# 1.1.1.1:777,
```

```
RP/0/RP0/CPU0:router(config-ext)# 1111:777
RP/0/RP0/CPU0:router(config-ext)# end-set
```

2. Configure a route policy with wildcards to refer to the extended community sets.

```
RP/0/RP0/CPU0:router(config)# route-policy WILDCARD_EXT_COMMUNITY_SET
RP/0/RP0/CPU0:router(config-rpl)# if extcommunity rt matches-any extcommunity-set* then
  pass else drop endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

This route policy configuration accepts routes with extended community set values as mentioned in the two extended community sets, and drops all other non-matching routes.

3. Commit your configuration.

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

This completes the configuration of routing policy with wildcards for extended community sets. For detailed information on extended community path sets, see [extcommunity-set, on page 497](#).

Use Wildcards for Route Distinguisher Sets

Use the following example to configure a routing policy with wildcards for route distinguisher sets.

1. Configure the route distinguisher sets in the global configuration mode.

```
RP/0/RP0/CPU0:router(config)# rd-set rd_set_demo
RP/0/RP0/CPU0:router(config-rd)# 10.0.0.1/8:77,
RP/0/RP0/CPU0:router(config-rd)# 10.0.0.2:888,
RP/0/RP0/CPU0:router(config-rd)# 65000:777
RP/0/RP0/CPU0:router(config-rd)# end-set
RP/0/RP0/CPU0:router(config)# rd-set rd_set_demo2
RP/0/RP0/CPU0:router(config-rd)# 20.0.0.1/7:99,
RP/0/RP0/CPU0:router(config-rd)# 4784:199
RP/0/RP0/CPU0:router(config-rd)# end-set
```

2. Configure a route policy with wildcards to refer to the route distinguisher set.

```
RP/0/RP0/CPU0:router(config)# route-policy use_rd_set
RP/0/RP0/CPU0:router(config-rpl)# if rd in rd-set* then set local-preference 100
RP/0/RP0/CPU0:router(config-rpl-if)# elseif rd in(10.0.0.2:888, 10.0.0.2:999)then set
  local-preference 300
RP/0/RP0/CPU0:router(config-rpl-elseif)# endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

3. Commit your configuration.

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

4. (Optional) Verify your configuration.

```
RP/0/RP0/CPU0:router(config)# show configuration
...
Building configuration...
!! IOS XR Configuration 0.0.0
!
rd-set rd_set_demo
  10.0.0.1/8:77,
  10.0.0.2:888,
```



```

        65000:777
end-set
!
!
rd-set rd_set_demo2
    20.0.0.1/7:99,
    4784:199
end-set
!

route-policy use_rd_set
    if rd in rd-set* then
        set local-preference 100
    elseif rd in (10.0.0.2:888, 10.0.0.2:999) then
        set local-preference 300
    endif
end-policy
!
end

```

This completes the configuration of routing policy with wildcards for route distinguisher sets. For more information on route distinguisher sets, see [rd-set](#), on page 502.

Use Wildcards for OSPF Area Sets

Use the following example to configure a routing policy with wildcards for OSPF area sets.

1. Configure the OSPF area set in the global configuration mode.

```

RP/0/RP0/CPU0:router(config)# ospf-area-set ospf_area_set_demo1
RP/0/RP0/CPU0:router(config-ospf-area)# 10.0.0.1,
RP/0/RP0/CPU0:router(config-ospf-area)# 3553
RP/0/RP0/CPU0:router(config-ospf-area)# end-set

RP/0/RP0/CPU0:router(config)# ospf-area-set ospf_area_set_demo2
RP/0/RP0/CPU0:router(config-ospf-area)# 20.0.0.2,
RP/0/RP0/CPU0:router(config-ospf-area)# 3673
RP/0/RP0/CPU0:router(config-ospf-area)# end-set

```

2. Configure a route policy with wildcards to refer to the OSPF area set.

```

RP/0/RP0/CPU0:router(config)# route-policy use_ospf_area_set
RP/0/RP0/CPU0:router(config-rpl)# if ospf-area in ospf-area-set* then set ospf-metric
200
RP/0/RP0/CPU0:router(config-rpl-if)# elseif ospf-area in( 10.0.0.1, 10.0.0.2 )then set
ospf-metric 300
RP/0/RP0/CPU0:router(config-rpl-elseif)# endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy

```

3. Commit your configuration.

```

RP/0/RP0/CPU0:router(config)# commit

```

4. (Optional) Verify your configuration.

```

RP/0/RP0/CPU0:router(config)# show configuration
Building configuration...
!! IOS XR Configuration 0.0.0
!
ospf-area-set ospf_area_set_demo1
    10.0.0.1,
    3553

```

```

end-set
!
!
ospf-area-set ospf_area_set_demo2
  20.0.0.2,
  3673
end-set
!

route-policy use_ospf_area_set
  if ospf-area in ospf-area-set* then
    set ospf-metric 200
  elseif ospf-area in (10.0.0.1, 10.0.0.2) then
    set ospf-metric 300
  endif
end-policy
!
end

```

This completes the configuration of routing policy with wildcards for OSPF area sets.

Translating Cisco IOS Route Maps to Cisco IOS XR Routing Policy Language: Example

RPL performs the same functions as route-maps. See *Converting Cisco IOS Configurations to Cisco IOS XR Configurations*.

VRF Import Policy Configuration: Example

This is a sample configuration for VRF import policy.

```

router bgp 100
address-family vpnv4 unicast
  vrf all
    source rt import-policy
  !

```

Additional References

The following sections provide references related to implementing RPL.

Related Documents

Related Topic	Document Title
Routing policy language commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Routing Policy Language Commands on Cisco IOS XR Software</i> module of the <i>Routing Command Reference for Cisco CRS Routers</i>

Related Topic	Document Title
Regular expression syntax	<i>Understanding Regular Expressions, Special Characters and Patterns</i> appendix in the <i>Cisco IOS XR Getting Started Guide for the Cisco CRS Router</i>

Standards

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

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

RFCs

RFCs	Title
RFC 1771	A Border Gateway Protocol 4 (BGP-4)
RFC 4360	BGP Extended Communities Attribute

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport



CHAPTER 12

Implementing Static Routes

This module describes how to implement static routes.

Static routes are user-defined routes that cause packets moving between a source and a destination to take a specified path. Static routes can be important if the Cisco IOS XR software cannot build a route to a particular destination. They are useful for specifying a gateway of last resort to which all unroutable packets are sent.



Note For more information about static routes on the Cisco IOS XR software and complete descriptions of the static routes commands listed in this module, see the [Related Documents, on page 597](#) section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the *Cisco IOS XR Commands Master List for the Cisco CRS Router*.

Feature History for Implementing Static Routes

Release	Modification
Release 2.0	This feature was introduced.
Release 3.3.0	Support for configuring static routes in static router configuration mode was added. The route command was replaced with the router static command. The route maximum command was replaced with the maximum path command. VPN routing and forwarding (VRF) support was added to the command syntax.
Release 3.4.0	No modification.
Release 3.5.0	IPv6 Provider Edge and IPv6 VPN Provider Edge over Multiprotocol Label Switching support was added.
Release 4.2.1	The Enhanced Object Tracking for IP Static feature was added.

- [Prerequisites for Implementing Static Routes, on page 584](#)
- [Restrictions for Implementing Static Routes, on page 584](#)
- [Information About Implementing Static Routes, on page 584](#)
- [How to Implement Static Routes, on page 587](#)
- [Configuration Examples, on page 594](#)

- [Where to Go Next, on page 597](#)
- [Additional References, on page 597](#)

Prerequisites for Implementing Static Routes

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.

Restrictions for Implementing Static Routes

These restrictions apply while implementing Static Routes:

- Static routing to an indirect next hop, (any prefix learnt through the RIB and may be more specific over the AIB), that is part of a local subnet requires configuring static routes in the global table indicating the egress interfaces as next hop. To avoid forward drop, configure static routes in the global table indicating the next-hop IP address to be the next hop.
- Generally, a route is learnt from the AIB in the global table and is installed in the FIB. However, this behavior will not be replicated to leaked prefixes. Because the AIB from the global table is not present in the VRF, the leaked FIB entry takes reference from the RIB rather than the same view as the global table, which also relies on the AIB. This could lead to inconsistencies in forwarding behavior.

Information About Implementing Static Routes

To implement static routes you need to understand the following concepts:

Static Route Functional Overview

Networking devices forward packets using route information that is either manually configured or dynamically learned using a routing protocol. Static routes are manually configured and define an explicit path between two networking devices. Unlike a dynamic routing protocol, static routes are not automatically updated and must be manually reconfigured if the network topology changes. The benefits of using static routes include security and resource efficiency. Static routes use less bandwidth than dynamic routing protocols, and no CPU cycles are used to calculate and communicate routes. The main disadvantage to using static routes is the lack of automatic reconfiguration if the network topology changes.

Static routes can be redistributed into dynamic routing protocols, but routes generated by dynamic routing protocols cannot be redistributed into the static routing table. No algorithm exists to prevent the configuration of routing loops that use static routes.

Static routes are useful for smaller networks with only one path to an outside network and to provide security for a larger network for certain types of traffic or links to other networks that need more control. In general, most networks use dynamic routing protocols to communicate between networking devices but may have one or two static routes configured for special cases.

Default Administrative Distance

Static routes have a default administrative distance of 1. A low number indicates a preferred route. By default, static routes are preferred to routes learned by routing protocols. Therefore, you can configure an administrative distance with a static route if you want the static route to be overridden by dynamic routes. For example, you could have routes installed by the Open Shortest Path First (OSPF) protocol with an administrative distance of 120. To have a static route that would be overridden by an OSPF dynamic route, specify an administrative distance greater than 120.

Directly Connected Routes

The routing table considers the static routes that point to an interface as “directly connected.” Directly connected networks are advertised by IGP routing protocols if a corresponding **interface** command is contained under the router configuration stanza of that protocol.

In directly attached static routes, only the output interface is specified. The destination is assumed to be directly attached to this interface, so the packet destination is used as the next hop address. The following example shows how to specify that all destinations with address prefix 2001:0DB8::/32 are directly reachable through interface GigabitEthernet 0/5/0/0:

```
RP/0/RP0/CPU0:router(config)# router static
RP/0/RP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-static-afi)# 2001:0DB8::/32 gigabitEthernet 0/5/0/0
```

Directly attached static routes are candidates for insertion in the routing table only if they refer to a valid interface; that is, an interface that is both up and has IPv4 or IPv6 enabled on it.

Recursive Static Routes

In a recursive static route, only the next hop is specified. The output interface is derived from the next hop. The following example shows how to specify that all destinations with address prefix 2001:0DB8::/32 are reachable through the host with address 2001:0DB8:3000::1:

```
RP/0/RP0/CPU0:router(config)# router static
RP/0/RP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-static-afi)# 2001:0DB8::/32 2001:0DB8:3000::1
```

A recursive static route is valid (that is, it is a candidate for insertion in the routing table) only when the specified next hop resolves, either directly or indirectly, to a valid output interface, provided the route does not self-recurse, and the recursion depth does not exceed the maximum IPv6 forwarding recursion depth.

A route self-recurses if it is itself used to resolve its own next hop. If a static route becomes self-recursive, RIB sends a notification to static routes to withdraw the recursive route.

Assuming a BGP route 2001:0DB8:3000::0/16 with next hop of 2001:0DB8::0104, the following static route would not be inserted into the IPv6 RIB because the BGP route next hop resolves through the static route and the static route resolves through the BGP route making it self-recursive:

```
RP/0/RP0/CPU0:router(config)# router static
RP/0/RP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-static-afi)# 001:0DB8::/32 2001:0DB8:3000::1
```

This static route is not inserted into the IPv6 routing table because it is self-recursive. The next hop of the static route, 2001:0DB8:3000:1, resolves through the BGP route 2001:0DB8:3000:0/16, which is itself a

recursive route (that is, it only specifies a next hop). The next hop of the BGP route, 2001:0DB8::0104, resolves through the static route. Therefore, the static route would be used to resolve its own next hop.

It is not normally useful to manually configure a self-recursive static route, although it is not prohibited. However, a recursive static route that has been inserted in the routing table may become self-recursive as a result of some transient change in the network learned through a dynamic routing protocol. If this occurs, the fact that the static route has become self-recursive will be detected and it will be removed from the routing table, although not from the configuration. A subsequent network change may cause the static route to no longer be self-recursive, in which case it is re-inserted in the routing table.

Fully Specified Static Routes

In a fully specified static route, both the output interface and next hop are specified. This form of static route is used when the output interface is multiaccess and it is necessary to explicitly identify the next hop. The next hop must be directly attached to the specified output interface. The following example shows a definition of a fully specified static route:

```
RP/0/RP0/CPU0:router(config)# router static
RP/0/RP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-static-afi)# 2001:0DB8::/32 GigEthernet0/0/0/0 2001:0DB8:3000::1
```

A fully specified route is valid (that is, a candidate for insertion into the routing table) when the specified interface, IPv4 or IPv6, is enabled and up.

Floating Static Routes

Floating static routes are static routes that are used to back up dynamic routes learned through configured routing protocols. A floating static route is configured with a higher administrative distance than the dynamic routing protocol it is backing up. As a result, the dynamic route learned through the routing protocol is always preferred to the floating static route. If the dynamic route learned through the routing protocol is lost, the floating static route is used in its place. The following example shows how to define a floating static route:

```
RP/0/RP0/CPU0:router(config)# router static
RP/0/RP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-static-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 210
```

Any of the three types of static routes can be used as a floating static route. A floating static route must be configured with an administrative distance that is greater than the administrative distance of the dynamic routing protocol because routes with smaller administrative distances are preferred.



Note By default, static routes have smaller administrative distances than dynamic routes, so static routes are preferred to dynamic routes.

Default VRF

A static route is always associated with a VPN routing and forwarding (VRF) instance. The VRF can be the default VRF or a specified VRF. Specifying a VRF, using the `vrf vrf-name` command, allows you to enter VRF configuration mode for a specific VRF where you can configure a static route. If a VRF is not specified, a default VRF static route is configured.

IPv4 and IPv6 Static VRF Routes

An IPv4 or IPv6 static VRF route is the same as a static route configured for the default VRF. The IPv4 and IPv6 address families are supported in each VRF.

How to Implement Static Routes

This section contains the following procedures:

Configure Static Route

Static routes are entirely user configurable and can point to a next-hop interface, next-hop IP address, or both. In the software, if an interface was specified, then the static route is installed in the Routing Information Base (RIB) if the interface is reachable. If an interface was not specified, the route is installed if the next-hop address is reachable. The only exception to this configuration is when a static route is configured with the permanent attribute, in which case it is installed in RIB regardless of reachability.



Note Currently, only default VRF is supported. VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families will be supported in a future release.

This task explains how to configure a static route.

SUMMARY STEPS

1. **configure**
2. **router static**
3. **vrf** *vrf-name*
4. **address-family** { **ipv4** | **ipv6** } { **unicast** | **multicast** }
5. *prefix mask* [**vrf** *vrf-name*] { *ip-address* | *interface-type interface-instance* } [*distance*] [**description text**] [**tag tag**] [**permanent**]
6. **commit**

DETAILED STEPS

Step 1 **configure**

Step 2 **router static**

Example:

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

Enters static route configuration mode.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-static)# vrf vrf_A
```

(Optional) Enters VRF configuration mode.

If a VRF is not specified, the static route is configured under the default VRF.

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

Example:

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

Enters address family mode.

Step 5 *prefix mask* [**vrf** *vrf-name*] { *ip-address* | *interface-type interface-instance* } [*distance*] [**description** *text*] [**tag** *tag*] [**permanent**]

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 10.0.0.0/8 172.20.16.6 110
```

Configures an administrative distance of 110.

- This example shows how to route packets for network 10.0.0.0 through to a next hop at 172.20.16.6 if dynamic information with administrative distance less than 110 is not available.

Step 6 **commit**

A default static route is often used in simple router topologies. In the following example, a route is configured with an administrative distance of 110.

```
configure
router static
address-family ipv4 unicast
0.0.0.0/0 2.6.0.1 110
end
```

Configuring a Static Route Under Multicast SAFI

This task explains how to configure a static route under multicast SAFI.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **vrf** *vrf-name*
4. **address-family** { **ipv4** | **ipv6** } **multicast**
5. **exit**
6. **router static**
7. **address-family** { **ipv4** | **ipv6** } { **unicast** | **multicast** } **topology** *topology-name*

8. `prefix mask [vrf vrf-name] [topology topology-name] { ip-address | type interface-path-id } [distance] [description text] [tag tag] [permanent]`
9. `commit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	<code>interface type interface-path-id</code> Example: <pre>RP/0/RP0/CPU0:router(config)# interface Loopback0</pre>	Enters interface configuration mode.
Step 3	<code>vrf vrf-name</code> Example: <pre>RP/0/RP0/CPU0:router(config-if)# vrf vrf1</pre>	(Optional) Enters VRF configuration mode. If a VRF is not specified, the static route is configured under the default VRF.
Step 4	<code>address-family { ipv4 ipv6 } multicast</code> Example: <pre>RP/0/RP0/CPU0:router(config-if)# address-family ipv4 multicast</pre>	Enters SAFI mode.
Step 5	<code>exit</code> Example: <pre>RP/0/RP0/CPU0:router(config-if)# exit</pre>	Exits interface configuration mode and enters global configuration mode.
Step 6	<code>router static</code> Example: <pre>RP/0/RP0/CPU0:router(config)#router static</pre>	Enters static route configuration mode.
Step 7	<code>address-family { ipv4 ipv6 } { unicast multicast } topology topology-name</code> Example: <pre>RP/0/RP0/CPU0:router(config-static)# address-family ipv4 unicast topology topol</pre>	Enters address family configuration mode and configures the topology to be associated with the interface.
Step 8	<code>prefix mask [vrf vrf-name] [topology topology-name] { ip-address type interface-path-id } [distance] [description text] [tag tag] [permanent]</code> Example: <pre>RP/0/RP0/CPU0:router(config-static-vrf-afi-topo)# 172.29.52.7/32 Loopback 0 110</pre>	Configures an administrative distance of 110.

	Command or Action	Purpose
Step 9	commit	

Configure Floating Static Route

This task explains how to configure a floating static route.

SUMMARY STEPS

1. **configure**
2. **router static**
3. **vrf vrf-name**
4. **address-family { ipv4 | ipv6 } { unicast | multicast }**
5. **prefix mask [vrf vrf-name] { ip-address | interface-type interface-instance } [distance] [description text] [tag tag] [permanent]**
6. **commit**

DETAILED STEPS

Step 1 **configure**

Step 2 **router static**

Example:

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

Enters static route configuration mode.

Step 3 **vrf vrf-name**

Example:

```
RP/0/RP0/CPU0:router(config-static)# vrf vrf_A
```

(Optional) Enters VRF configuration mode.

If a VRF is not specified, the static route is configured under the default VRF.

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

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf)# address family ipv6 unicast
```

Enters address family mode.

Step 5 **prefix mask [vrf vrf-name] { ip-address | interface-type interface-instance } [distance] [description text] [tag tag] [permanent]**

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
```

Configures an administrative distance of 201.

Step 6 **commit**

A floating static route is often used to provide a backup path if connectivity fails. In the following example, a route is configured with an administrative distance of 201.

```
configure
router static
address-family ipv6 unicast
2001:0DB8::/32 2001:0DB8:3000::1 201
end
```

Configure Static Routes Between PE-CE Routers

This task explains how to configure static routing between PE-CE routers.



Note VRF fallback is not supported with IPv6 VPN Provider Edge (6VPE).

SUMMARY STEPS

1. **configure**
2. **router static**
3. **vrf** *vrf-name*
4. **address-family** { **ipv4** | **ipv6** } { **unicast** | **multicast** }
5. *prefix mask* [**vrf** *vrf-name*] { *ip-address* | *interface-type interface-path-id* } [*distance*] [**description** *text*] [**tag** *tag*] [**permanent**]
6. **commit**

DETAILED STEPS

Step 1 **configure**

Step 2 **router static**

Example:

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

Enters static route configuration mode.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-static)# vrf vrf_A
```

(Optional) Enters VRF configuration mode.

If a VRF is not specified, the static route is configured under the default VRF.

Step 4 `address-family { ipv4 | ipv6 } { unicast | multicast }`

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf)# address family ipv6 unicast
```

Enters address family mode.

Step 5 `prefix mask [vrf vrf-name] { ip-address | interface-type interface-path-id } [distance] [description text] [tag tag] [permanent]`

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
```

Configures an administrative distance of 201.

Step 6 `commit`

In the following example, a static route between PE and CE routers is configured, and a VRF is associated with the static route:

```
configure
router static
vrf vrf_A
address-family ipv4 unicast
0.0.0.0/0 2.6.0.2 120
end
```

Change Maximum Number of Allowable Static Routes

This task explains how to change the maximum number of allowable static routes.

Before you begin



Note The number of static routes that can be configured on a router for a given address family is limited by default to 4000. The limit can be raised or lowered using the **maximum path** command. Note that if you use the **maximum path** command to reduce the configured maximum allowed number of static routes for a given address family below the number of static routes currently configured, the change is rejected. In addition, understand the following behavior: If you commit a batch of routes that would, when grouped, push the number of static routes configured above the maximum allowed, the first *n* routes in the batch are accepted. The number previously configured is accepted, and the remainder are rejected. The *n* argument is the difference between the maximum number allowed and number previously configured.

SUMMARY STEPS

1. `configure`

2. **router static**
3. **maximum path** { **ipv4** | **ipv6** } *value*
4. **commit**

DETAILED STEPS

Step 1 **configure**

Step 2 **router static**

Example:

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

Enters static route configuration mode.

Step 3 **maximum path** { **ipv4** | **ipv6** } *value*

Example:

```
RP/0/RP0/CPU0:router(config-static)# maximum path ipv4 10000
```

Changes the maximum number of allowable static routes.

- Specify IPv4 or IPv6 address prefixes.
- Specify the maximum number of static routes for the given address family. The range is from 1 to 140000.
- This example sets the maximum number of static IPv4 routes to 10000.

Step 4 **commit**

Configuring a static route to point at interface null 0 may be used for discarding traffic to a particular prefix. For example, if it is required to discard all traffic to prefix 2001:0DB8:42:1/64, the following static route would be defined:

```
configure
router static
address-family ipv6 unicast
2001:0DB8:42:1::/64 null 0
end
```

Associate VRF with a Static Route

This task explains how to associate a VRF with a static route.

SUMMARY STEPS

1. **configure**
2. **router static**

3. **vrf** *vrf-name*
4. **address-family** { **ipv4** | **ipv6** } { **unicast** | **multicast** }
5. *prefix mask* [**vrf** *vrf-name*] {**next-hop** *ip-address* | *interface-name* } {*path-id*} [*distance*] [**description** *text*] [**tag** *tag*] [**permanent**]
6. **commit**

DETAILED STEPS

Step 1 **configure**

Step 2 **router static**

Example:

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

Enters static route configuration mode.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-static)# vrf vrf_A
```

Enters VRF configuration mode.

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

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf)# address family ipv6 unicast
```

Enters address family mode.

Step 5 *prefix mask* [**vrf** *vrf-name*] {**next-hop** *ip-address* | *interface-name* } {*path-id*} [*distance*] [**description** *text*] [**tag** *tag*] [**permanent**]

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
```

Configures an administrative distance of 201.

Step 6 **commit**

Configuration Examples

This section provides the following configuration examples:

Configuring Traffic Discard: Example

Configuring a static route to point at interface null 0 may be used for discarding traffic to a particular prefix. For example, if it is required to discard all traffic to prefix 2001:0DB8:42:1::/64, the following static route would be defined:

```
configure
router static
address-family ipv6 unicast
2001:0DB8:42:1::/64 null 0
end
```

Configuring a Fixed Default Route: Example

A default static route is often used in simple router topologies. In the following example, a route is configured with an administrative distance of 110.

```
configure
router static
address-family ipv4 unicast
0.0.0.0/0 2.6.0.1 110
end
```

Configuring a Floating Static Route: Example

A floating static route is often used to provide a backup path if connectivity fails. In the following example, a route is configured with an administrative distance of 201.

```
configure
router static
address-family ipv6 unicast
2001:0DB8::/32 2001:0DB8:3000::1 201
end
```

Configure Native UCMP for Static Routing

In a network where traffic is load balanced on two or more links, configuring equal metrics on the links would create Equal Cost Multipath (ECMP) next hops. Because the bandwidth of the links is not taken into consideration while load balancing, the higher bandwidth links are underutilized. To avoid this problem, you can configure Unequal Cost Multipath (UCMP), either locally (local UCMP), or natively (native UCMP) so that the higher bandwidth links carry traffic in proportion to the capacity of the links. UCMP supports IPv4 and IPv6 static VRF routes.

Local UCMP: All static routes are configured with the same link metrics. The static IGP calculates the load metric based on the bandwidth of the links and load balances the traffic across the links. However, local UCMP does not consider bandwidth while load balancing across links that are closer to the destination (multiple hops away).

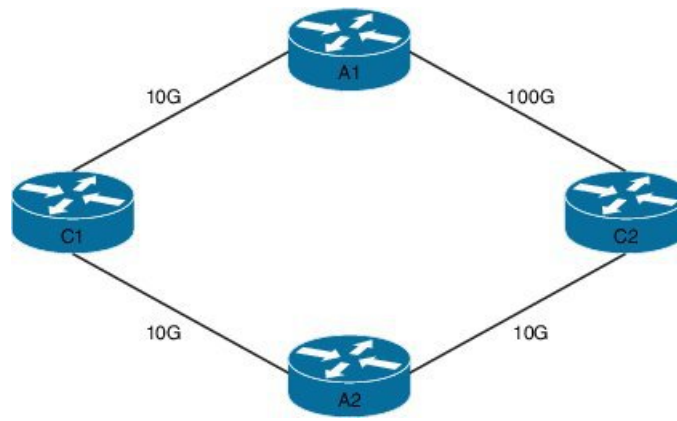
Native UCMP: Static routes over higher bandwidth links are configured with lower link metrics so that they are preferred to routes over lower bandwidth links. The static IGP calculates the load metric based on the bandwidth of the links and determines the percentage of traffic going out of the higher and lower bandwidth

links. By matching the configured link metrics with end-to-end available bandwidth, native UCMP is able to effectively load balance traffic across links that are closer to the destination (multiple hops away).

Configuration Example

Consider the topology in the following figure. For load balancing traffic out of Router A1, if local UCMP is used, then both 10G and 100G links will have equal link metrics. The static IGP decides to send more traffic out of the 100G link because of the higher load metric. However, for load balancing traffic out of Router A2, local UCMP works only on links to Routers C1 and C2. For load balancing traffic from Router C1 to Router A1 and Router C2 to Router A1, native UCMP is preferred. As a result, local UCMP is used only on single hop destinations, and native UCMP is used for multi-hop destinations.

Figure 26: Unequal Cost Multipath for Static Routing



To configure UCMP for static routing, use the following steps:

1. Enter the global configuration mode.

```
RP/0/0/CPU0:Router# configure
```

2. Enter the static routing mode.

```
RP/0/0/CPU0:Router(config)# router static
```

3. Configure UCMP with load metric for IPv4 or IPv6 address family.

```
RP/0/0/CPU0:Router(config-static)# address-family ipv4 unicast
RP/0/0/CPU0:Router(config-static-afi)# 10.10.10.1/32 GigabitEthernet 0/0/0/1 metric 10
```

In this example, we have configured UCMP for IPv4 address family. To configure UCMP for IPv6 address family, use the following sample configuration.

```
RP/0/0/CPU0:Router(config-static)# address-family ipv6 unicast
RP/0/0/CPU0:Router(config-static-afi)# 10:10::1/64 GigabitEthernet 0/0/0/1 metric 10
```

4. Exit the static configuration mode and commit your configuration.

```
RP/0/0/CPU0:Router(config-static-afi)# exit
RP/0/0/CPU0:Router(config-static)# exit
RP/0/0/CPU0:Router(config)# commit
Fri Feb 19 06:16:33.164 IST
RP/0/0/CPU0:Feb 19 06:16:34.273 : ipv4_static[1044]:
%ROUTING-IP_STATIC-4-CONFIG_NEXTHOP_ETHER_INTERFACE :
Route for 10.10.10.1 is configured via ethernet interface
```

Repeat this procedure on all routers that need to be configured with UCMP.

Configuring a Static Route Between PE-CE Routers: Example

In the following example, a static route between PE and CE routers is configured, and a VRF is associated with the static route:

```
configure
router static
vrf vrf_A
address-family ipv4 unicast
0.0.0.0/0 2.6.0.2 120
end
```

Where to Go Next

For additional information about static routes, routing protocols, and RIB, consult the following publications:

- *Implementing and Monitoring RIB on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco CRS Routers*
- *Implementing BGP on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco CRS Routers*
- *Implementing EIGRP on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco CRS Routers*
- *Implementing IS-IS on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco CRS Routers*
- *Implementing MPLS Layer 3 VPN on Cisco IOS XR Software* in *Cisco IOS XR Multiprotocol Label Switching Configuration Guide*
- *Implementing OSPF on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco CRS Routers*
- *Implementing OSPFv3 on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco CRS Routers*
- *RIB Commands on Cisco IOS XR Software* in *Routing Command Reference for Cisco CRS Routers*
- *Implementing RIP on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco CRS Routers*

Additional References

The following sections provide references related to implementing Static Routes.

Related Documents

Related Topic	Document Title
Static routes commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Static Routing Commands</i> in <i>Routing Command Reference for Cisco CRS Routers</i>
MPLS Layer 3 VPN configuration: configuration concepts, task, and examples	<i>MPLS Configuration Guide for the Cisco CRS Routers</i>

Standards

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

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

RFCs

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

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access more content.	http://www.cisco.com/techsupport



CHAPTER 13

Implementing RCMD

This module describes how to implement RCMD.

Feature History for Implementing RCMD

Release	Modification
Release 4.2.0	This feature was introduced.

- [Route Convergence Monitoring and Diagnostics, on page 599](#)
- [Route Convergence Monitoring and Diagnostics Database Monitoring, on page 600](#)
- [Configuring Route Convergence Monitoring and Diagnostics, on page 600](#)
- [Route Convergence Monitoring and Diagnostics Prefix Monitoring, on page 603](#)
- [Route Convergence Monitoring and Diagnostics OSPF Type 3/5/7 Link-state Advertisements Monitoring, on page 603](#)
- [Enabling RCMD Monitoring for IS-IS Prefixes, on page 603](#)
- [Enable RCMD Monitoring for OSPF Prefixes, on page 604](#)
- [Enabling RCMD Monitoring for Type 3/5/7 OSPF LSAs, on page 605](#)
- [Enabling RCMD Monitoring for IS-IS Prefixes: Example, on page 606](#)
- [Enabling RCMD Monitoring for OSPF Prefixes: Example, on page 606](#)
- [Enabling RCMD Monitoring for Type 3/5/7 OSPF LSAs: Example, on page 607](#)

Route Convergence Monitoring and Diagnostics

Route Convergence Monitoring and Diagnostics (RCMD) is a mechanism to monitor OSPF and ISIS convergence events, gather details about the SPF runs and time taken to provision routes and LDP labels across all LCs on the router.

RCMD is a tool that collects and reports data related to routing convergence. Highlights of the RCMD mechanism are:

- Lightweight and always-on using route flow markers across routing components (all nodes & MC).
- Tracks most convergence events and all routes affected by them.
- Provides within-router view with statistics and time-lines on per convergence event basis.
- Measurements against time-line/SLA and triggers specified EEM actions on excess.

- 'On the router' reports via CLI/XML interface.
- Each RCMD enabled router provides a digest of convergence data.

The events that are monitored and reported by RCMD are:

- OSPF and IS-IS SPF events (default VRF only).
- Add/delete of specific external or inter-area/level prefixes.
- IGP flooding propagation delays for LSA/LSP changes.

RCMD runs in two modes:

- Monitoring—detecting events and measuring convergence.
- Diagnostics—additional (debug) information collection for 'abnormal' events.

Route Convergence Monitoring and Diagnostics Database Monitoring

The Route Convergence Monitoring and Diagnostics (RCMD) database monitoring enables LFA computation report in Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS). The detailed LFA computation report is displayed as a separate report and is not associated with any event.

Configuring Route Convergence Monitoring and Diagnostics

Perform these tasks to configure route convergence monitoring and diagnostics:

SUMMARY STEPS

1. **configure**
2. **router-convergence**
3. **collect-diagnostics** *location*
4. **event-buffer-size** *number*
5. **max-events-stored** *number*
6. **monitoring-interval** *minutes*
7. **node** *node-name*
8. **protocol**
9. **priority**
10. **disable**
11. **leaf-network** *number*
12. **threshold** *value*
13. **storage-location**
14. **diagnostics** *directory-path-name*
15. **diagnostics-size**
16. **reports** *directory-path-name*

17. reports-size

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router-convergence Example: RP/0/RP0/CPU0:router (config)#router-convergence	Enters configure Router Convergence Monitoring and Diagnostics (rcmd) configuration mode.
Step 3	collect-diagnostics <i>location</i> Example: RP/0/RP0/CPU0:router (config-rcmd)#collect-diagnostics 0/3/CPU0	Configures to collect diagnostics on specified node.
Step 4	event-buffer-size <i>number</i> Example: RP/0/RP0/CPU0:router (config-rcmd)#event-buffer-size 100	Sets event buffer size (as number of events) for storing event traces .
Step 5	max-events-stored <i>number</i> Example: RP/0/RP0/CPU0:router (config-rcmd)#max-events-stored 10	Sets maximum number of events to be stored in the server.
Step 6	monitoring-interval <i>minutes</i> Example: RP/0/RP0/CPU0:router (config-rcmd)#monitoring-interval 120	Sets interval (in minutes) to collect logs.
Step 7	node <i>node-name</i> RP/0/RP0/CPU0:router (config-rcmd)#node	Configures parameters for a specified node.
Step 8	protocol Example: RP/0/RP0/CPU0:router (config-rcmd)#protocol ISIS RP/0/RP0/CPU0:router (config-rcmd-PROTO)#	Specifies the protocol for which to configure RCMD parameters. <ul style="list-style-type: none"> • ISIS-Select ISIS to configure parameters related to ISIS protocol • OSPF-Select OSPF to configure parameters related to OSPF protocol
Step 9	priority Example: RP/0/RP0/CPU0:router (config-rcmd-PROTO)#priority critical RP/0/RP0/CPU0:router (config-rcmd-PROTO-PRIO)#	Sets priority for monitoring of route convergence for the specified protocol. <ul style="list-style-type: none"> • Critical-Set to monitor route convergence for critical priority routes • High-Set to monitor route convergence for high priority routes

	Command or Action	Purpose
		<ul style="list-style-type: none"> • Medium-Set to monitor route convergence for medium priority routes • Low-Set to monitor route convergence for low priority routes
Step 10	disable Example: RP/0/RP0/CPU0:router (config-rcmd- <i>proto-prio</i>) #disable	Disables the monitoring of route convergence for specified priority.
Step 11	leaf-network <i>number</i> Example: RP/0/RP0/CPU0:router (config-rcmd- <i>proto-prio</i>) #leaf-network 100	Enables leaf network monitoring. Specify a maximum number of leaf networks to be monitored. Range for maximum number is 10-100.
Step 12	threshold <i>value</i> Example: RP/0/RP0/CPU0:router (config-rcmd- <i>proto-prio</i>) #threshold 1000	Specifies threshold value for convergence in milliseconds. Select a threshold value from the range. Range is 0-4294967295 milliseconds
Step 13	storage-location Example: RP/0/RP0/CPU0:router (config-rcmd) #storage-location RP/0/RP0/CPU0:router (config-rcmd- <i>store</i>) #	Sets the absolute directory path for storing diagnostic reports.
Step 14	diagnostics <i>directory-path-name</i> Example: RP/0/RP0/CPU0:router (config-rcmd- <i>store</i>) #diagnostics /disk0:/rcmd	Specifies the absolute directory path for storing diagnostic reports. Set a directory-path-name. Example: /disk0:/rcmd/ or <ftp-location>/rcmd/
Step 15	diagnostics-size Example: RP/0/RP0/CPU0:router (config-rcmd- <i>store</i>) # diagnostics-size 8	Specify a maximum size for the diagnostics directory. Set the size in %. Range is 5%-80%.
Step 16	reports <i>directory-path-name</i> Example: RP/0/RP0/CPU0:router (config-rcmd- <i>store</i>) #reports /disk0:/rcmd	Specifies the absolute directory path for storing reports. Set a directory-path-name. Example: /disk0:/rcmd/ or <ftp-location>/rcmd/
Step 17	reports-size Example: RP/0/RP0/CPU0:router (config-rcmd- <i>store</i>) #reports-size 8	Specify a maximum size for the reports directory. Set the size in %. Range is 5%-80%.

Route Convergence Monitoring and Diagnostics Prefix Monitoring

The Route Convergence Monitoring and Diagnostics (RCMD) prefix monitoring feature enables convergence monitoring for specific individual prefixes in Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS) Interior Gateway Protocols (IGP). In IGP, when the route information is created, the prefix is verified against the configured prefix-list. If the prefix is found to be monitored, it is marked for monitoring and information about each prefix change event is captured. The RCMD prefix monitoring individually monitors specific prefixes on each RCMD enabled router in the network. A maximum of 10 prefixes can be monitored. Individual prefix monitoring compliments the probes enabled at customer network edges to monitor connectivity and availability of specific service end-points.

The RCMD prefix monitoring for IS-IS prefixes is enabled by configuring the **prefix-list** command under Router IS-IS monitor-convergence configuration mode. The RCMD prefix monitoring for OSPF prefixes is enabled by configuring the **prefix-list** command under Router OSPF monitor-convergence configuration mode.

For individual prefix monitoring, the prefixes are marked before those appear for the route calculation so that the monitoring does not affect the convergence of OSPF or ISIS routes.

Route Convergence Monitoring and Diagnostics OSPF Type 3/5/7 Link-state Advertisements Monitoring

The Route Convergence Monitoring and Diagnostics (RCMD) OSPF type 3/5/7 link-state advertisements (LSA) monitoring feature flags and differentiates the LSAs during the monitoring of LSAs. A change in route for type 3/5/7 LSAs has to be monitored. During the route calculation, if the route source appears to be type 3/5/7 LSAs and the route change is an add or delete action, then those prefixes have to be monitored. RCMD monitors all deletion of available paths (a purge operation) and addition of the first path (a restoration operation) for all type 3/5/7 LSAs. The OSPF type 3/5/7 LSAs are monitored and reported on a individual prefix basis. However, a modify operation that involves a change in paths not affecting reachability as a whole, is not monitored. Although all prefixes are logged for reporting, the convergence tracking is rate-limited for the first 10 prefixes that are affected in an SPF run.

The RCMD OSPF type 3/5/7 LSA monitoring is enabled by configuring the **track-external-routes** and **track-summary-routes** under Router OSPF monitor-convergence configuration mode.

Enabling RCMD Monitoring for IS-IS Prefixes

Perform this task to enable individual prefix monitoring for IS-IS prefixes.

Before you begin

To enable monitoring of individual prefixes, first configure a prefix-list using the **{ipv4 | ipv6} prefix-list** command. Then, use this prefix list with the **prefix-list** command.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **address-family** {*ipv4* | *ipv6*} [**unicast** | **multicast**]
4. **monitor-convergence**
5. **prefix-list** *prefix-list-name*
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router isis <i>instance-id</i> Example: RP/0/RP0/CPU0:router(config)#router isis isp	Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.
Step 3	address-family { <i>ipv4</i> <i>ipv6</i> } [unicast multicast] Example: RP/0/RP0/CPU0:router(config-isis)#address-family ipv6 unicast	Enter the IS-IS address-family configuration mode.
Step 4	monitor-convergence Example: RP/0/RP0/CPU0:router(config-isis-af)#monitor-convergence	Enables route convergence monitoring for IS-IS protocol.
Step 5	prefix-list <i>prefix-list-name</i> Example: RP/0/RP0/CPU0:router(config-isis-af-rcmd)#prefix-list isis_monitor	Enables individual prefix monitoring for IS-IS prefixes.
Step 6	commit	

Enable RCMD Monitoring for OSPF Prefixes

Perform this task to enable individual prefix monitoring for OSPF prefixes.

Before you begin

To enable monitoring of individual prefixes, first configure a prefix-list using the {*ipv4* | *ipv6*} **prefix-list** command. Then, use this prefix list with the **prefix-list** command.

SUMMARY STEPS

1. **configure**
2. **router ospf** *ospf-process-name*
3. **monitor-convergence**

4. **prefix-list** *prefix-list-name*
5. **commit**

DETAILED STEPS

Step 1 **configure**

Step 2 **router ospf** *ospf-process-name*

Example:

```
RP/0/RP0/CPU0:router(config)#router ospf 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Step 3 **monitor-convergence**

Example:

```
RP/0/RP0/CPU0:router(config-ospf)#monitor-convergence
```

Enables OSPF route convergence monitoring.

Step 4 **prefix-list** *prefix-list-name*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-af-rcmd)#prefix-list ospf_monitor
```

Enables individual prefix monitoring for OSPF prefixes.

Step 5 **commit**

Enabling RCMD Monitoring for OSPF Prefixes: Example

This example shows how to enable RCMD monitoring for individual OSPF prefixes:

```
ipv6 prefix-list ospf_monitor
 10 permit 2001:db8::/32
!
router ospf 100
 monitor-convergence
  prefix-list ospf_monitor
```

Enabling RCMD Monitoring for Type 3/5/7 OSPF LSAs

Perform this task to enable RCMD monitoring for type 3/5/7 OSPF LSAs.

SUMMARY STEPS

1. **configure**
2. **router ospf** *100*
3. **track-external-routes**

4. `track-summary-routes`
5. `commit`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>configure</code>	
Step 2	<code>router ospf 100</code> Example: RP/0/RP0/CPU0:router(config)#router ospf 100	Enables OSPF routing for the specified routing process and places the router in router configuration mode.
Step 3	<code>track-external-routes</code> Example: RP/0/RP0/CPU0:router(config-ospf-af-rcmd)#track-external-routes	Enables tracking of external (Type-3/5/7) LSAs prefix monitoring.
Step 4	<code>track-summary-routes</code> Example: RP/0/RP0/CPU0:router(config-ospf-af-rcmd)#track-summary-routes	Enables tracking of summary (inter-area) routes monitoring
Step 5	<code>commit</code>	

Enabling RCMD Monitoring for IS-IS Prefixes: Example

This example shows how to monitor RCMD prefix monitoring for individual IS-IS prefixes:

```
ipv6 prefix-list isis_monitor
 10 permit 2001:db8::/32
!
router isis isp
 address-family ipv6 unicast
  monitor-convergence
  prefix-list isis_monitor
```

Enabling RCMD Monitoring for OSPF Prefixes: Example

This example shows how to enable RCMD monitoring for individual OSPF prefixes:

```
ipv6 prefix-list ospf_monitor
 10 permit 2001:db8::/32
!
router ospf 100
 monitor-convergence
  prefix-list ospf_monitor
```

Enabling RCMD Monitoring for Type 3/5/7 OSPF LSAs: Example

This example shows how to enable tracking of prefix monitoring for OSPF external LSAs and summary routes:

```
router ospf 100
 monitor-convergence
  track-external-routes
  track-summary-routes
```




CHAPTER 14

Implementing Data Plane Security

The data plane security (DPSec) feature prevents traffic injection from external sources into a LISP VPN. DPSec relies on the integrity of the routing locator (RLOC) network which is built using unicast reverse path forwarding (URPF) support.

In order to enable LISP shared mode segmentation without incurring the overhead of authentication and encryption, the DPSec feature uses a mechanism called Source RLOC Decapsulation Filtering that enforces URPF on the network. The URPF configured on the network disseminates lists of acceptable RLOCs, traffic from which will already have been proofed by URPF. This makes it impossible to spoof the source RLOC address of LISP control and data packets. DPSec feature uses the list of valid encapsulation sources for each EID instance to filter LISP data packets during decapsulation at xTRs and PxTRs.

Feature History for Data Plane Security

Release 5.3.0	This feature was introduced.
------------------	------------------------------

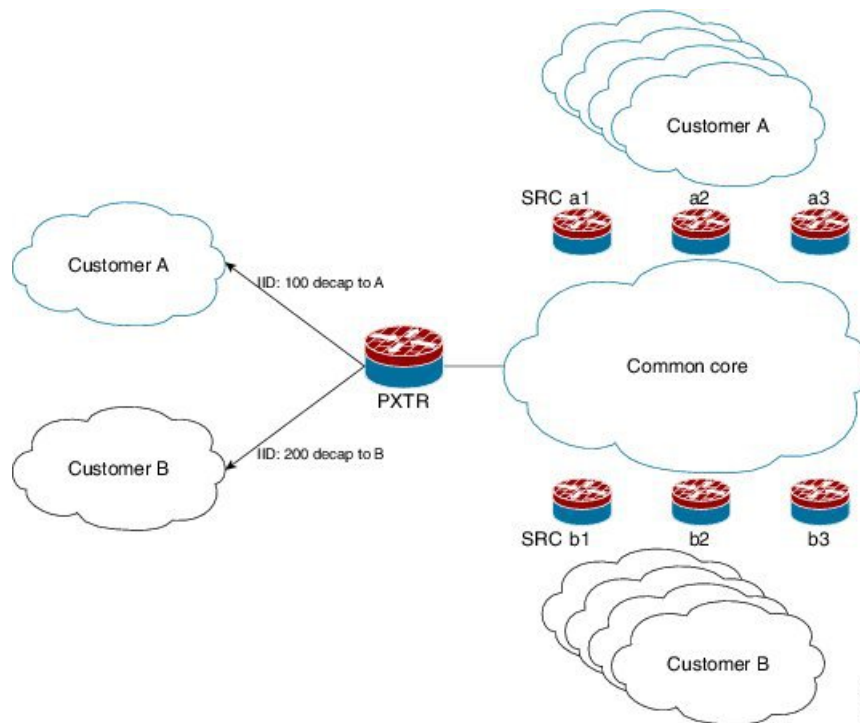
- [Information about Data Plane Security, on page 609](#)
- [How to Implement Data Plane Security, on page 614](#)
- [Additional References, on page 624](#)

Information about Data Plane Security

The LISP Data Plane Security feature ensures that only traffic from within a LISP VPN can be decapsulated into the VPN. In order to understand data plane security you must be familiar with the following features and concepts it supports:

Source RLOC Decapsulation Filtering

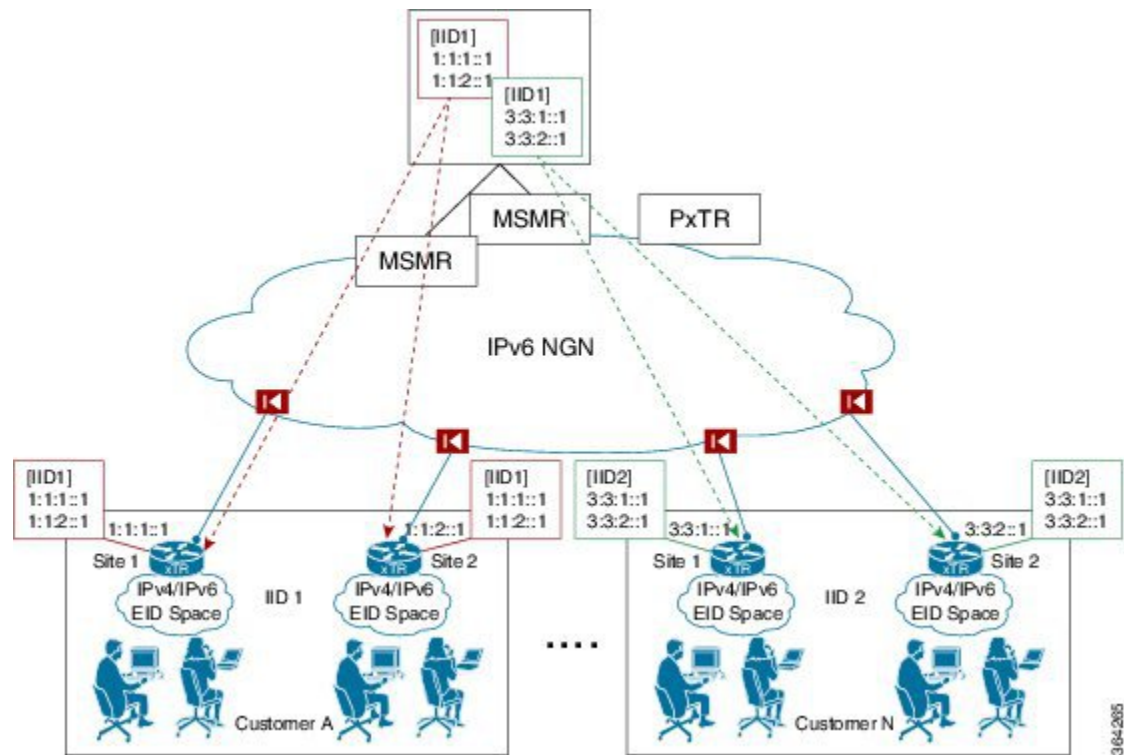
This illustration shows blue and black customer networks using LISP EID instance ID (IID) 100 and 200, respectively, over a shared common RLOC core. When decapsulating LISP data packets, the PxTR validates that packets carrying instance ID 100 have a source (SRC) RLOC in the encapsulation header of either a1, a2 or a3. Similarly, for instance ID 200 the PxTR validates that the RLOC source is b1, b2 or b3.



LISP encapsulated data packets that do not carry a valid source RLOC are dropped. The combination of RLOC space URPF enforcement and source RLOC-based decapsulation filtering ensures that it not possible for a source that is not member of a tenant VPN to inject traffic into the VPN.

EID Instance Membership Distribution

To deploy the source RLOC filtering solution, an automated mechanism is required to push the list of valid RLOCs through the mapping system to the boxes performing decapsulation. This function is performed by the Map-Servers. The Map-Servers construct the EID instance ID RLOC membership list using the RLOC information in the received mapping records in Map-Register messages. The complete list is then pushed out to all the xTRs and PxTRs that must decapsulate packets for the VPN identified by the EID instance ID.



In the example, the Map-Servers build a separate VPN (EID instance) membership list for each customer and then push the contents of the list out. The two xTRs for customer A each register their site RLOCs. They each receive back from the Map-Server the complete list of RLOCs of all the xTRs for customer A. The received list is used to filter decapsulated traffic and enforce the data plane security.

When PxTRs are being used (for example to provide internet connectivity to the VPN) then the xTRs participating in the VPN must accept and decapsulate the LISP data packets sent by the PxTRs. The RLOC addresses used by the PxTRs have to be included in the EID instance membership list communicated to the xTRs by the Map-Server. The PxTRs do not register EID prefixes with the Map-Server that the Map-Server can use to discover the PxTR RLOCs. Those RLOCs will have to be manually configured on the Map-Server.

The EID instance membership lists built by Map-Servers are only useful to boxes participating in the VPN. As an added security measure, the Map-Server will only communicate the contents of the membership list for an EID instance to xTRs and PxTRs that are members of that VPN.

Map-Server Membership Gleaning and Distribution

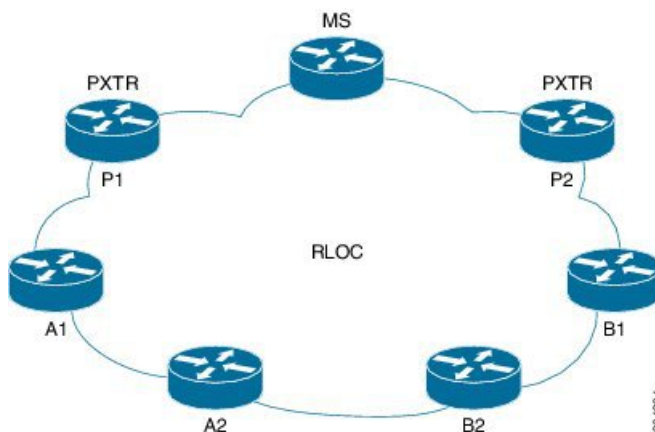
A LISP Map-Server is responsible for tracking the per EID instance membership and distributing it to (P)xTRs. Use the **map-server rloc members distribute** command to enable this functionality. The command configures the Map-Server to:

- Build a list of RLOC addresses using Map-Registrations and configuration from which to accept reliable transport sessions.
- Accept TCP connections from (P)xTRs in above list.
- Glean and maintain per EID instance RLOC membership from received Map-Register messages.

- Serve EID instance membership requests received over the reliable transport sessions from (P)xTRs and distribute membership information.

The per EID instance membership list that the MS gleans from received registrations can be extended or completely overridden through the **map-server rloc members {add | override}** configuration command. The command allows the user to extend the discovered xTR RLOC membership with PxTR RLOC addresses. The extended membership list is used to determine whether to allow a membership request that is received over a reliable transport session. Only requests from xTRs that have registrations in an EID instance are allowed. The extended membership list is then pushed to decapsulating devices implementing the data plane security feature that will then be able to accept encapsulated packets sent by both valid xTRs and PxTRs.

To prevent unauthorized attempts to establish TCP connections with the Map-Server, a list of allowed locators from which to accept connections is built. The list contains the RLOC addresses of the registering xTRs as well as the RLOC addresses configured in membership list extensions. Note that there is a single list from which to accept connections per RLOC address family (it is not EID instance specific).



As an example consider the network in the above figure with two VPNs. VPNs A and B each have two xTRs A1/A2 and B1/B2 respectively. The membership of VPN A is extended on the MS through the “map-server rloc members add ...” configuration to include PxTR RLOC address P1. The membership of VPN B is extended to include PxTR RLOC address P2. The resulting lists maintained by the MS are:

- EID instance 1 (VPN A) membership: A1, A2, P1
- EID instance 2 (VPN B) membership: B1, B2, P2
- Locators from which to accept TCP session: A1, A2, P1, B1, B2, P2

The Map-Server may receive an EID instance membership request for one or more EID instances through each established reliable transport session. PxTRs will typically request the membership of multiple instances through the single session that they establish with the MS. The Map-Server must provide full membership refreshes and incremental updates for each of the accepted requests.

When a membership request is received by an MS and the peer (P)xTR originating the request is not a member of the EID instance to which the request pertains, then the MS will reject the request and return a membership NACK message to the (P)xTR. Note that such an event may occur during normal operation as the TCP session and membership request from an xTR may be received before the corresponding Map-Register message that places it in the EID instance membership. If after an EID instance membership request has been accepted by an MS, the requesting (P)xTR is removed from the EID instance membership because of a registration expiration or configuration change, then the MS will send a Membership-NACK message to the (P)xTR to indicate that it is no longer receiving membership updates for that instance.

When a Map-Server restarts, it must first discover and rebuild the EID instance membership lists before serving membership requests. Specifically the MS must hold off sending any membership refresh end messages for EID instances that do not have a complete membership list. On the MS the LISP control plane will wait to receive registrations before considering the membership list complete. The following conditions must be met:

- At least one registration period has elapsed (one minute) after the first registration was received and one of the following conditions holds:
 - No accept-more-specific site EID prefix configuration exists for the EID instance and registrations for all the configured EID prefixes have been received.
 - Three registration periods have elapsed from the time that the first registration was received.
 - No registrations have been received and three registration periods have elapsed from the time that the LISP control plane restarted.

You can manage membership distribution on the Map-Server using the **show lisp site rloc members** command in the EXEC configuration mode.

[How to Implement Data Plane Security, on page 614](#) provides procedural details.

Decapsulation Filtering on (P)xTRs

The source RLOC decapsulation RLOC filtering feature is enabled on a (P)xTR through the **decapsulation filter rloc source** command. After the feature is enabled, the (P)xTR only allows the decapsulation of LISP data packets carrying a source RLOC that is allowed by the filter. When the feature is first enabled, if the filter is based on the auto-discovery of the EID instance membership from the Map-Servers then traffic will be dropped until a reliable transport connection is established with the Map-Servers and the membership is received.

(P)xTR Membership Discovery

A (P)xTR that is configured for data plane source RLOC filtering with membership auto-discovery for one or more EID instances through the **decapsulation filter rloc source members** configuration, attempt to establish a reliable transport session with each of the configured Map-Servers for those instances. A single reliable transport session is initiated with each Map-Server over which the membership for one or more EID instances is communicated. The auto-discovered membership lists is extended to form the source rloc filter through the **locator-set** option of the **decapsulation filter rloc source** command. The membership lists for an EID instance discovered through each of the Map-Servers are merged together with the contents of the configured locator-set and used to define the data plane source RLOC. The Map-Server only accepts incoming reliable transport connections from RLOC addresses that have first successfully registered an EID prefix. An xTR only attempts to establish a connection after it receives a Map-Notify acknowledging that its registration was successful. In order to request EID instance membership services for a specific instance ID at least one EID prefix for that instance must have been successfully registered.

Once the connection with a Map-Server is established the (P)xTR sends a Membership-Request message for each of the EID instances that have the Map-Server in their configuration. Received Membership-Add and Membership-Delete messages update the EID instance membership database on the (P)xTR.

To rebuild its EID instance membership database, the (P)xTR issues a Membership-Refresh-Request message as soon as the Map-Server indicates that it is willing to provide membership services through a Membership-ACK message. The (P)xTR maintains an epoch for each discovered membership entry. When a Membership-Refresh-Start message is received from a Map-Server, the (P)xTR increments the epoch it

maintains for the Map-Server and EID instance combination, thus flagging the existing membership state as stale. Subsequent Membership-Add messages received during the refresh update the epoch of the corresponding entries. When the Membership-Refresh-End message is received, the (P)xTR sweeps the membership entries for the EID instance received from the Map-Server deleting the ones carrying an old epoch that have not been updated during the refresh.

Filter Communication to Forwarding

The LISP control plane uses the RIB opaque facility for communicating information through the RIB, all the way to all FIB instances as part of table distribution. Messages are defined to:

- Convey the filter enablement state on a per RLOC AF and EID instance granularity
- Convey RLOC filter entries

TCP-based Reliable Transport Sessions

LISP uses TCP-based sessions between the xTRs and Map-Servers for EID instance membership distribution. The reliable transport session supports (using TCP port 4342) establishing of an active or passive session, with the xTR taking the active role and the Map-Server the passive role. Sessions are accepted only from valid RLOCs from the Map-Server side based on source RLOC filtering. The number of concurrent TCP connections that can be supported varies on a per OS and platform basis. Some security considerations that you must be take into account:

- The number of xTRs that a Map-Server can cater for is limited by the number of TCP sessions that a platform can establish and maintain. This will determine the number of VPN customers that a Map-Server can host. Horizontal scaling is achieved by dividing VPN customers between multiple Map-Servers.
- All the xTRs belonging to the same VPN must register with the same Map-Server. You cannot have VPNs with a larger number of xTRs than the Map-Server TCP session scale limit.
- Session authentication of the initial deliverable relies on the integrity of the RLOC network and only filters TCP sessions using the source address of packets.

For additional details on TCP-based reliable transport session such as Session Establishment, Reliable Transport Message Format, Keep-alive Message, Error Notification Message, see <http://tools.ietf.org/id/draft-kouvelas-lisp-reliable-transport-00.txt>.

How to Implement Data Plane Security

This section contains the following procedures:

Enable Source RLOC-based Decapsulation Filtering

To configure an xTR or Proxy-xTR to download decapsulation filter lists for source validation when decapsulating LISP packets, use the **decapsulation filter source** command in the lisp configuration mode.

When an (P)ETR decapsulates LISP packets, this occurs without consideration of the LISP packet outer header source address. In networking environments where the source address can be trusted, it may be desired to consider the source address of the LISP packet prior to decapsulation. By configuring the decapsulation filter source command on (P)xTRs, a device will establish a TCP-based reliable transport session with its

Map-Server(s) and download and use filter list(s) when decapsulating LISP packets. Either or both of the **members** or **locator-set** keywords must be specified.

When the **members** keyword is specified the xTR will attempt to establish a reliable transport (TCP) sessions with the configured map-servers to automatically obtain the registered RLOC membership list. When a **locator-set** is specified the filtering will be performed against the locators that are configured within the locator-set. When both the locator-set and the "members" keyword are specified then the configured locators and the automatically discovered ones will be merged and the resulting list used to filter decapsulated packets.



Note

- A (P)xTR normally communicates with multiple Map-Servers. However, in the event that all reliable transport session goes down, any existing (possibly stale) filter list will remain in use during a small window of time (several minutes), during which time the (P)xTR tries to re-establish the session(s) with the MS and refresh its membership.
- If no filter list can be downloaded, or the existing list times out, packets will be dropped. (fail closed.)
- If the xTR changes RLOCs (via DHCP for example), as soon as the RLOC is changed, the registration with the Map-Server is updated and the new registered RLOC is pushed to all "members" of this IID/VPN (event-driven).

Before you begin

Ensure that the following pre-requisites are met:

- On an xTR, the TCP-based reliable transport session is established only after the UDP-based (normal) Map-Registration process successfully completes.
- On a PxTR, since this device does not (normally) register with a Map-Server, a 'stub' (fake) Map-Registration configuration must be added to allow the establishment of the reliable transport session and the download of any filter lists. The Map-Server requires the PETER RLOC(s) to be included in a map-server rloc members modify-discovered add command to permit this session establishment.

SUMMARY STEPS

1. **configure**
2. **router lisp**
3. **exit**
4. **locator-set** *name IP_address*
5. **eid-table** { **default** | [**vrf** *vrf_name*] } **instance-id** *instance_id*
6. **address-family** { **ipv4** | **ipv6** } **unicast**
7. **etr map-server** *IP_address* { **key** [**clear** | **encrypted**] **LINE** | **proxy-reply** }
8. **itr map-resolver** *map-resolver-address*
9. **map-cache** *destination-EID-prefix/prefix-length* { **action** { **drop** | **map-request** | **native-forward** } | **locator** *locator-address* **priority** *priority_value* | **weight** *weight_value*
10. **database-mapping** *EID-prefix/prefixlength locator* **locator-set** *site* **priority** *priority* **weight** *weight*
11. **exit**
12. **decapsulation filter rloc source** [**locator-set** *locator_set_name*] [**members**]
13. **locator-table** *name* [**default** | **vrf** *vrf_name*]

14. commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure	
Step 2	router lisp Example: RP/0/RP0/CPU0:router(config)# router lisp	Enables LISP for the specified routing instance, and places the router in Locator and ID Separation Protocol (LISP) configuration mode.
Step 3	exit Example: RP/0/RP0/CPU0:routerRP/0/0/CPU0:ios(config-lisp-afi)#exit	Returns the router to LISP configuration mode.
Step 4	locator-set name IP_address Example: RP/0/RP0/CPU0:router(config-lisp)#locator-set loc_sh1_vrf1 202.1.0.1	Configure a named locator set site, and specifies the RLOC IP address of Loopback or other Egress Tunnel Router (ETR) interfaces.
Step 5	eid-table { default [vrf vrf_name] } instance-id instance_id Example: RP/0/RP0/CPU0:router(config-lisp)#eid-table default instance-id <IID-A>	Selects the default (global) routing table or the specified VRF table for association with the configured instance ID.
Step 6	address-family { ipv4 ipv6 } unicast Example: RP/0/RP0/CPU0:router(config-lisp-afi)# address-family ipv4 unicast	Specifies the IPv4 or IPv6 address family, and enters address family configuration mode. <ul style="list-style-type: none">• This example specifies the unicast IPv4 address family.
Step 7	etr map-server IP_address { key [clear encrypted] LINE proxy-reply } Example: RP/0/RP0/CPU0:router(config-lisp-afi)#etr map-server 204.1.0.1 key encrypted lisp	Specifies the options related to the etr map-server (MS) such as locator and authentication key.
Step 8	itr map-resolver map-resolver-address Example: RP/0/RP0/CPU0:router(config-lisp-afi)#itr map-resolver 204.1.0.1	Configures an IPv4 or IPv6 locator address of the LISP Map-Resolver to be used by the ITR Map-Requests for IPv4 EID-to-RLOC mapping resolution.

	Command or Action	Purpose
Step 9	<p>map-cache <i>destination-EID-prefix / prefix-length</i> { action { drop map-request native-forward } locator <i>locator-address</i> priority <i>priority_value</i> weight <i>weight_value</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-lisp-afi)#map-cache 12.2.0.0/24 map-request RP/0/RP0/CPU0:router(config-lisp-afi)#map-cache 102.2.0.0/24 map-request RP/0/RP0/CPU0:router(config-lisp-afi)#map-cache 103.2.0.0/24 map-request</pre>	Configures a static IPv4 EID-to-RLOC or static IPv6 EID-to-RLOC mapping relationship and its associated traffic policy, or statically configure the packet handling behavior associated with a destination IPv4 EID-prefix or a destination IPv6 EID-prefix.
Step 10	<p>database-mapping <i>EID-prefix/prefixlength locator locator-set site</i> priority <i>priority</i> weight <i>weight</i></p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-lisp-afi)#database-mapping 11.2.0.0/24 201.1.0.1 priority 1 weight 100</pre>	Configures an EID-to-RLOC mapping relationship and its associated traffic policy for this LISP site. Note Repeat this step until all EID-to-RLOC mappings within this eid-table vrf and instance ID for the LISP site are configured.
Step 11	<p>exit</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-lisp-afi)#exit</pre>	Returns the router to LISP configuration mode.
Step 12	<p>decapsulation filter rloc source [locator-set <i>locator_set_name</i>] [members]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-lisp)#decapsulation filter rloc source member locator-set loc_sh1_vrf1</pre>	Enables the source RLOC based decapsulation filtering feature. <ul style="list-style-type: none"> • The members keyword enables the establishment of a reliable transport (TCP) session with configured Map-Server(s), and the download of the decapsulation filer list maintained by the Map-Server(s) and the download of the decapsulation filer list maintained by the Map-Server(s) • The locator-set keyword is used, the prefixes named in the locator-set are used, if included alone, or added to the (downloaded) dynamic list when used in conjunction with the member keyword.
Step 13	<p>locator-table <i>name</i> [default vrf <i>vrf_name</i>]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router(config-lisp)#locator-table vrf 1</pre>	Associate a virtual routing and forwarding (VRF) table through which the routing locator address space is reachable to a router Locator ID Separation Protocol (LISP) instantiation.

	Command or Action	Purpose
Step 14	commit	

Example

In this example, an xTR is configured to establish a reliable transport session with the Map-Server at 204.1.0.1, download the decapsulation filter list (in this case for IID 1002), and source-check all LISP-encapsulated packets using this filter list prior to decapsulation.

```
router lisp
 address-family ipv4 unicast
 !
 locator-set loc_sh1_vrf1
 202.1.0.1
 203.1.0.1
 !
 eid-table vrf sh1_vrf2 instance-id 1002
 address-family ipv4 unicast
  etr map-server 204.1.0.1 key encrypted lisp
  etr
  itr map-resolver 204.1.0.1
  itr
  map-cache 12.2.0.0/24 map-request
  map-cache 102.2.0.0/24 map-request
  map-cache 103.2.0.0/24 map-request
  database-mapping 11.2.0.0/24 201.1.0.1 priority 1 weight 100
  database-mapping 101.2.0.0/24 201.1.0.1 priority 1 weight 100
 !
 decapsulation filter rloc source member locator-set
  loc_sh1_vrf1
 !
 locator-table default
```

Create, Maintain and Distribute Decapsulation Filter Lists

A Map-Server can be configured to dynamically create, maintain, and distribute decapsulation filter lists, on a per instance-ID basis, to appropriate LISP devices using the map-server rloc members distribute command in site configuration mode. When configured:

- The Map-Server allows the establishment of TCP-based LISP reliable transport sessions with appropriate xTRs
- The Map-Server creates/maintains lists (per-IID) of LISP site RLOCs (per-IID) based on RLOC addresses of registered LISP sites
- The Map-Server pushes/updates filters lists over the reliable transport mechanism to established devices



Note

- Data plane security is enabled by the use of the “map-server rloc members distribute” command. The optional command “map-server rloc members modified-discovered [add | override] is used to append to or override the dynamically maintained RLOC filter list.
- This feature is used in conjunction with the decapsulation filter rloc source command, configured on (P)xTR devices which are performing the decapsulation

This example shows how you can configure the Map-Server to create reliable transport sessions with specific LISP sites, to dynamically create, maintain, and distribute decapsulation filter lists.

```
router lisp
 locator-set PxTR_set
  2001:DB8:E:F::2
 exit
 !
eid-table vrf 1001 instance-id 1001
 map-server rloc members modify-discovered add locator-set PxTR_set
 exit
 !
---<skip>---
 !
 map-server rloc members distribute
 !
```

Add or Override Decapsulation Filter List

When a Map-Server is configured to dynamically create, maintain, and distribute a decapsulation filter list, the decapsulation filter list can be added to or overridden by using the `map-server rloc members modify-discovered` command in EID-table configuration mode. Uses may include:

- When a PxTR is included in the architecture, the Pitr LISP-encapsulates packets to an ETR – and the ETR must therefore include the Pitr RLOC in its decapsulation filter list. Since PitrS do not register with Map-Servers, their RLOCs are not automatically included in the decapsulation filter list and must be added via configuration using this command.
- A PETER can also be configured to filter upon decapsulation, but again, because a PETER does not register with a Map-Server, it needs a way to obtain the decapsulation filter list. The **add** form of this command includes the mechanisms to establish the reliable transport session with the Map-Server for obtaining the decapsulation filter list on the PETER.
- For diagnostic/troubleshooting reasons, it may be useful to (temporarily) override the entire decapsulation filter list.



Note The add function must be included in order for a PETER to “fake registers” with the Map-Server to obtain the decapsulation filter list. The inclusion of the PETER RLOC in this command allows the PETER to establish the reliable transport session.

In this example, the Map-Server is configured to create reliable transport sessions with specific LISP sites, to dynamically create, maintain, and distribute decapsulation filter lists. It has also been configured to modify the dynamically created filter list (comprised of registered site RLOC addresses) with the statically configured PxTR IPv6 RLOC address of 2001:db8:e:f::2.

```
router lisp
 locator-set PxTR_set
  2001:DB8:E:F::2
 exit
 !
eid-table vrf 1001 instance-id 1001
 map-server rloc members modify-discovered add locator-set PxTR_set
 ipv4 route-export site-registration
 exit
 !
```

```

----<skip>----
!
map-server rloc members distribute
!

```

Reset LISP TCP Reliable Transport Session

To reset the LISP TCP reliable transport session between an xTR and an MS use the **clear lisp vrf** command in the EXEC mode.

SUMMARY STEPS

1. **clear lisp vrf** *VRF_name* **session** {*peer_address* | *}

DETAILED STEPS

	Command or Action	Purpose
Step 1	clear lisp vrf <i>VRF_name</i> session { <i>peer_address</i> *} Example: RP/0/0/CPU0:ios#clear lisp vrf test session *	When a peer-address is specified the TCP connection to that peer will be cleared. When the "*" option is specified all LISP reliable transport sessions will be cleared.

Verify Data Plane Security Configurations

Perform this task to verify data plane security configurations:

SUMMARY STEPS

1. **show lisp session**
2. **show lisp site** [*instance-id* *EID instance-ID*] **rloc members** [**registrations** [*rloc-addr*]]
3. **show lisp vrf** *vrf_name* **session** [*peer_address*]
4. **show lisp decapsulation filter**
5. **show cef vrf** [*locator-vrf*] *address_family* **lisp decapsulation** [*instance-id* *EID-instance-ID*] **detail location** *RLOC-facing LC*
6. **show controllers np struct LISP-INSTANCE-HASH detail all-entries** [**all** | *np*] **location** *RLOC-facing LC*

DETAILED STEPS

	Command or Action	Purpose
Step 1	show lisp session Example: R11-MSMR#show lisp session Sessions for VRF default, total: 8, established: 7 Peer State Up/Down In/Out Users 2001:DB8:A:1::2 Up 00:04:13 2/7 2	On a LISP device, to display a current list of reliable transport (TCP) sessions, use the show lisp session command in the EXEC configuration mode. In this example, reliable transport LISP sessions are displayed on the Map-Server. In the output, seven sessions are established and one session is in the Init state (the decapsulation filter rloc source member command had not been applied at that site; the session was not established).

	Command or Action	Purpose
	<pre> 2001:DB8:A:2::2 Up 00:04:13 2/7 2 2001:DB8:A:3::2 Up 00:03:53 2/7 2 2001:DB8:B:1::2 Up 00:04:04 2/6 2 2001:DB8:B:2::2 Init never 0/0 1 2001:DB8:C:1::2 Up 00:03:55 2/6 2 2001:DB8:C:2::2 Up 00:03:54 2/6 2 2001:DB8:E:F::2 Up 00:04:04 6/19 4 R11-MSMR# </pre>	
Step 2	<p>show lisp site [instance-id <i>EID instance-ID</i>] rloc members [registrations [<i>rloc-addr</i>]]</p> <p>Example:</p> <pre> R114-MSMR#show lisp site rloc members LISP RLOC membership for EID table default (IID 0), 5 entries RLOC Origin Valid 1.2.3.4 config Yes 10.0.1.2 registration Yes 10.0.2.2 config & registration Yes 13:12::1 config Yes 2001:DB8:2:3::2 registration Yes </pre>	<p>To display the gleaned and configured EID instance membership use the show lisp site command in the EXEC configuration mode. The 'origin' column in the output shows whether the RLOC member has been manually configured, automatically gleaned from received registrations, or both. The 'valid' column shows whether the RLOC is a valid member that is distributed to (P)xTRs. A listed RLOC may not be valid if it is gleaned from registrations, but the 'override' option is used in the 'modify-discovered' configuration and the specified locator-set does not include the RLOC. When the optional 'registrations' keyword is specified the command displays the list of registrations contributing to a membership entry.</p>
Step 3	<p>show lisp vrf <i>vrf_name</i> session [peer_address]</p> <p>Example:</p> <pre> On xTR: RP/0/RSP1/CPU0:VKG-1#sh lisp vrf default session Sessions for VRF default, total: 1, established: 1 Peer State Up/Down In/Out Users 204.1.0.1 Up 06:49:05 0/1 1 RP/0/RSP1/CPU0:VKG-1# On MSMR: sh lisp vrf default session Sessions for VRF default, total: 2, established: 2 Peer State Up/Down In/Out Users 201.1.0.1 Up 06:48:49 1/0 0 202.1.0.1 Up 06:48:36 2/0 0 RP/0/RSP0/CPU0:VKG-4# </pre>	

	Command or Action	Purpose																																
Step 4	<p>show lisp decapsulation filter</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:lisp9-a9k-1#show lisp eid-table se2 decapsulation filter LISP decapsulation filter for EID table vrf se2 (IID 16777212), 5 entries Source RLOC Added by 22:22::10 MS 190::190 33:33::20 MS 190::190 88:88::30 MS 190::190 99:99::30 MS 190::190 110:110::40 MS 190::190 RP/0/RSP0/CPU0:lisp9-a9k-1#</pre>	<p>On a LISP device, to display decapsulation filter-related data for the selected source RLOCs, use the show lisp decapsulation filter command in the EXEC configuration mode. In this example, decapsulation filter information is displayed on an (P)xTR for Instance-ID (IID 16777212). In this output, five source RLOC addresses are defined, and all members of this list were defined within the list provided by the reliable transport session with the Map-Server.</p>																																
Step 5	<p>show cef vrf [locator-vrf] address_family lisp decapsulation [instance-id EID-instance-ID] detail location RLOC-facing LC</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:lisp9-a9k-1#show cef ipv6 lisp decapsulation instance-id 16777212 loc 0/0/cpu0</pre> <p>Number of EID tables handling LISP payload received in this table: 3</p> <p>Transport LISP ipv6 packets received in VRF: default, Instance ID: 16777212</p> <pre> Payload IPv4 is : decapsulated Payload IPv6 is : decapsulated Payload switched in VRF : se2 (0xe000001d/0xe080001d) H/W driver signalled : active Binding in retry : no</pre> <p>Source RLOC Prefix Filter : enabled</p> <pre> Lookup statistics (s/w) : Misses : 8 Matches (historic) : 0 H/W driver signalled : active Binding in retry : no Platform space : allocated</pre> <table border="1"> <thead> <tr> <th>Len</th> <th>Prefix</th> <th>Action</th> <th>Matches</th> </tr> </thead> <tbody> <tr> <td>128</td> <td>22:22::10</td> <td>accept</td> <td>0 h/w</td> </tr> <tr> <td></td> <td>[active, plt space]</td> <td></td> <td></td> </tr> <tr> <td>128</td> <td>33:33::20</td> <td>accept</td> <td>0 h/w</td> </tr> <tr> <td></td> <td>[active, plt space]</td> <td></td> <td></td> </tr> <tr> <td>128</td> <td>88:88::30</td> <td>accept</td> <td>0 h/w</td> </tr> <tr> <td></td> <td>[active, plt space]</td> <td></td> <td></td> </tr> <tr> <td>128</td> <td>99:99::30</td> <td>accept</td> <td>0 h/w [active, plt space]</td> </tr> </tbody> </table>	Len	Prefix	Action	Matches	128	22:22::10	accept	0 h/w		[active, plt space]			128	33:33::20	accept	0 h/w		[active, plt space]			128	88:88::30	accept	0 h/w		[active, plt space]			128	99:99::30	accept	0 h/w [active, plt space]	<p>In this example, decapsulation filter summary information is displayed on an (P)xTR.</p>
Len	Prefix	Action	Matches																															
128	22:22::10	accept	0 h/w																															
	[active, plt space]																																	
128	33:33::20	accept	0 h/w																															
	[active, plt space]																																	
128	88:88::30	accept	0 h/w																															
	[active, plt space]																																	
128	99:99::30	accept	0 h/w [active, plt space]																															

	Command or Action	Purpose
	<pre>128 110:110::40 accept 0 h/w [active, plt space]</pre>	
<p>Step 6</p>	<p>show controllers np struct LISP-INSTANCE-HASH detail all-entries [all np] location RLOC-facing LC</p> <p>Example:</p> <pre>RP/0/RSP0/CPU0:lisp9-a9k-1#show controllers np struct LISP-INSTANCE-HASH detail all-entries np0 location 0/0/cpu0 Node: 0/0/CPU0: ----- NP: 0 Struct 114: LISP_INSTANCE_HASH_STR (maps to uCode Str=79) Struct is a LOGICAL entity inside a shared PHYSICAL resource Reserved Entries: Logical 0, Physical 0 Used Entries: Logical 79, Physical 79 Max Entries: Logical 86016, Physical 86016 Entries Shown: Logical 79 ----- Entry 1: >> Key: 4affffff 00000099 00990000 00000000 00000000 0030 Size: 22 Mask: ffffffff ffffffff ffffffff ffffffff ffffffff ffff Size: 22 Result: 51000000 1e000000 Size: 8 Entry 2: >> Key: 4976adf1 1c005858 0e1e0000 00000000 00000000 0000 Size: 22 Mask: ffffffff ffffffff ffffffff ffffffff ffffffff ffff Size: 22 Result: 51000000 1c000000 Size: 8 Entry 3: >> Key: 4976adf1 1c006e6e 0e280000 00000000 00000000 0000 Size: 22 Mask: ffffffff ffffffff ffffffff ffffffff ffffffff ffff Size: 22 Result: 51000000 1c000000 Size: 8 Entry 4: >> Key: 41000000 20002121 0b140000 00000000 00000000 0000 Size: 22 Mask: ffffffff ffffffff ffffffff ffffffff ffffffff ffff Size: 22 Result: 51000000 1f000000 Size: 8 Entry 5: >> Key: 4afffffc 00000099 00990000 00000000 00000000 0030 Size: 22 Mask: ffffffff ffffffff ffffffff ffffffff ffffffff ffff Size: 22 Result: 51000000 1d000000 Size: 8 Entry 6: >></pre>	

Command or Action	Purpose
<pre> Key: 4a0003e8 19000033 00330003 00000000 00000000 0020 Size: 22 Mask: ffffffff ffffffff ffffffff ffffffff fffffff ffff Size: 22 Result: 51000000 19000000 Size: 8 Entry 7: >> <Snipped> End NP Show Structure Display </pre>	

Additional References

The following sections provide references related to implementing LISP data plane security:

Related Documents

Related Topic	Document Title
LISP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Routing Command Reference for Cisco CRS Routers</i>
LISP Configuration Guide, Cisco IOS Release	IP Routing: LISP Configuration Guide, Cisco IOS Release 15M&T

Standards

Standards	Title
draft-kouvelas-lisp-reliable-transport-00.txt	<i>LISP Reliable Transport</i> by C. Cassar, I. Kouvelas and D. Lewis
RFC 6830	<i>Locator/ID Separation Protocol (LISP)</i>
RFC 6832	<i>Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites</i>
RFC 6833	<i>Locator/ID Separation Protocol (LISP) Map-Server Interface</i>

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport