Routing Configuration Guide for Cisco NCS 6000 Series Routers, IOS XR Release 7.4.x

First Published: 2021-07-01

Last Modified: 2020-01-22

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 NONINFRINGEMENT 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.

The documentation set for this product strives to use bias-free language. For purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on standards documentation, or language that is used by a referenced third-party product.

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: https://www.cisco.com/c/en/us/about/legal/trademarks.html. 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)

© 2021 Cisco Systems, Inc. All rights reserved.

CONTENTS

PREFACE

Preface xxi

Changes to This Document xxi

Communications, Services, and Additional Information xxi

CHAPTER 1

New and Changed Routing Features 1

New and Changed Routing Features 1

CHAPTER 2

Implementing BGP 3

Prerequisites for Implementing BGP 4

Information About Implementing BGP 4

BGP Functional Overview 4

BGP Router Identifier 4

BGP Default Limits 5

BGP Enhanced Multipath Selection 6

BGP Next Hop Tracking 8

Next Hop as the IPv6 Address of Peering Interface 9

Scoped IPv4 Table Walk 10

Reordered Address Family Processing 10

New Thread for Next-Hop Processing 10

show, clear, and debug Commands 10

Autonomous System Number Formats in BGP 10

2-byte Autonomous System Number Format 11

4-byte Autonomous System Number Format 11

as-format Command 11

BGP Configuration 11

Configuration Modes 11

Neighbor Submode 12
Configuration Templates 13
Template Inheritance Rules 14
Viewing Inherited Configurations 18
No Default Address Family 22
Neighbor Address Family Combinations 23
Routing Policy Enforcement 23
Table Policy 24
Update Groups 24
BGP Update Generation and Update Groups 25
BGP Update Group 25
BGP Cost Community 25
How BGP Cost Community Influences the Best Path Selection Process 25
Cost Community Support for Aggregate Routes and Multipaths 26
Influencing Route Preference in a Multiexit IGP Network 27
Adding Routes to the Routing Information Base 28
BGP Best Path Algorithm 28
Comparing Pairs of Paths 29
Order of Comparisons 31
Best Path Change Suppression 31
Administrative Distance 32
Route Dampening 33
Minimizing Flapping 34
BGP Routing Domain Confederation 34
BGP Route Reflectors 34
Remotely Triggered Null Route Filtering with RPL Next-hop Discard Configuration 37
Configuring Destination-based RTBH Filtering 38
Verification 39
Default Address Family for show Commands 40
BGP Keychains 40
BGP Session Authentication and Integrity using TCP Authentication Option Overview 41
Master Key Tuples 41
Configure BGP Session Authentication and Integrity using TCP Authentication Option 41
BGP Nonstop Routing 43

BGP Best-External Path 45
BGP Local Label Retention 46
BGP Over GRE Interfaces 46
Command Line Interface (CLI) Consistency for BGP Commands 46
BGP Additional Paths 46
Persistent Loadbalancing 47
BGP Selective Multipath 48
BFD Multihop Support for BGP 50
BGP Multi-Instance and Multi-AS 50
BGP Prefix Independent Convergence for RIB and FIB 51
BGP Update Message Error Handling 52
BGP Attribute Filtering 52
BGP Attribute Filter Actions 52
BGP Link-State 53
BGP VRF Dynamic Route Leaking 55
Recent Prefixes Events and Trace Support 56
How to Implement BGP 59
Enabling BGP Routing 59
Configuring Multiple BGP Instances for a Specific Autonomous System 62
Configuring a Routing Domain Confederation for BGP 63
Resetting an eBGP Session Immediately Upon Link Failure 65
Logging Neighbor Changes 65
Adjusting BGP Timers 65
Changing the BGP Default Local Preference Value 66
Configuring the MED Metric for BGP 67
Configuring BGP Weights 68
Tuning the BGP Best-Path Calculation 70
Indicating BGP Back-door Routes 71
Configuring Aggregate Addresses 72
Redistributing iBGP Routes into IGP 74
Redistributing Prefixes into Multiprotocol BGP 75
Configuring BGP Route Dampening 77
Applying Policy When Updating the Routing Table 78
Setting BGP Administrative Distance 79

Configuring a BGP Neighbor Group and Neighbors 80
Configuring a Route Reflector for BGP 83
Configuring BGP Route Filtering by Route Policy 85
Configuring BGP Attribute Filtering 86
Configuring BGP Next-Hop Trigger Delay 87
Disabling Next-Hop Processing on BGP Updates 88
Configuring BGP Community and Extended-Community Advertisements 90
Configuring the BGP Cost Community 92
Configuring Software to Store Updates from a Neighbor 95
Configuring Keychains for BGP 96
Disabling a BGP Neighbor 98
Resetting Neighbors Using BGP Inbound Soft Reset 99
Resetting Neighbors Using BGP Outbound Soft Reset 100
Resetting Neighbors Using BGP Hard Reset 100
Clearing Caches, Tables, and Databases 101
Displaying System and Network Statistics 101
Displaying BGP Process Information 103
Monitoring BGP Update Groups 104
Configuring BGP Nonstop Routing 105
Configuring BGP Additional Paths 106
Configuring VRF Dynamic Route Leaking 108
Configuration Examples for Implementing BGP 109
Enabling BGP: Example 109
Displaying BGP Update Groups: Example 111
BGP Neighbor Configuration: Example 111
BGP Confederation: Example 112
BGP Route Reflector: Example 114
BGP Nonstop Routing Configuration: Example 114
Primary Backup Path Installation: Example 114
VRF Dynamic Route Leaking Configuration: Example 114
Flow-tag propagation 115
Restrictions for Flow-Tag Propagation 115
Configuring Destination-Based Flow-Tag Propagation 116
Where to Go Next 118

Additional References 118

CHAPTER 3 **Implementing BGP Flowspec** 121

BGP Flow Specification 121

Limitations 122

BGP Flowspec Conceptual Architecture 122

Information About Implementing BGP Flowspec 123

Flow Specifications 123

Supported Matching Criteria and Actions 124

Traffic Filtering Actions 127

BGP Flowspec Client-Server (Controller) Model and Configuration 129

How to Configure BGP Flowspec 130

Enable Flowspec on BGP Side 131

Configure a Class Map 132

Define Policy Map 133

Link Flowspec to PBR Policies 135

Verify BGP Flowspec 137

Preserving Redirect Nexthop 139

Validate BGP Flowspec 140

Disabling BGP Flowspec 141

Disable Flowspec Redirect and Validation 142

Configuration Examples for Implementing BGP Flowspec 143

Flowspec Rule Configuration 143

Drop Packet Length 144

Remark DSCP 145

Additional References for BGP Flowspec 145

CHAPTER 4 **Implementing BFD** 147

Prerequisites for Implementing BFD 147

Restrictions for Implementing BFD 148

Information About BFD 149

Differences in BFD in Cisco IOS XR Software and Cisco IOS Software 149

BFD Modes of Operation 149

BFD Packet Information 150

BFD Source and Destination Ports 150
BFD Packet Intervals and Failure Detection 151
Priority Settings for BFD Packets 155
BFD for IPv4 155
BFD for IPv6 156
BFD on Bundled VLANs 156
BFD Over Member Links on Link Bundles 156
Overview of BFD State Change Behavior on Member Links and Bundle Status 157
BFD for MultiHop Paths 159
Setting up BFD Multihop 159
BFD IPv6 Multihop 159
BFD over MPLS Traffic Engineering LSPs 159
BFD over Bundle and BFD over Logical Bundle 160
BFD over Bundle 161
Bidirectional Forwarding Detection over Logical Bundle 162
BFD Object Tracking 163
How to Configure BFD 164
BFD Configuration Guidelines 164
Configuring BFD Under a Dynamic Routing Protocol or Using a Static Route 164
Enabling BFD on a BGP Neighbor 164
Enabling BFD for OSPF on an Interface 166
Enabling BFD for OSPFv3 on an Interface 168
Configuring BFD on Bundle Member Links 170
Prerequisites for Configuring BFD on Bundle Member Links 170
Specifying the BFD Destination Address on a Bundle 170
Enabling BFD Sessions on Bundle Members 171
Configuring the Minimum Thresholds for Maintaining an Active Bundle 172
Configuring BFD Packet Transmission Intervals and Failure Detection Times on a Bundle 173
Configuring Allowable Delays for BFD State Change Notifications Using Timers on a Bundle 175
Enabling Echo Mode to Test the Forwarding Path to a BFD Peer 176
Overriding the Default Echo Packet Source Address 176
Specifying the Echo Packet Source Address Globally for BFD 177
Specifying the Echo Packet Source Address on an Individual Interface or Bundle 177
Configuring BFD Session Teardown Based on Echo Latency Detection 179

Delaying BFD Session Startup Until Verification of Echo Path and Latency 180
Disabling Echo Mode 181
Disabling Echo Mode on a Router 182
Disabling Echo Mode on an Individual Interface 183
Minimizing BFD Session Flapping Using BFD Dampening 184
Enabling and Disabling IPv6 Checksum Support 185
Enabling and Disabling IPv6 Checksum Calculations for BFD on a Router 185
Enabling and Disabling IPv6 Checksum Calculations for BFD on an Individual Interface or Bundle 186
Clearing and Displaying BFD Counters 187
Configuring BFD over MPLS Traffic Engineering LSPs 188
Enabling BFD Parameters for BFD over TE Tunnels 188
Configuring BFD Bring up Timeout 189
Configuring BFD Dampening for TE Tunnels 190
Configuring Periodic LSP Ping Requests 192
Configuring BFD at the Tail End 193
Configuring BFD over LSP Sessions on Line Cards 194
Configuring BFD Object Tracking: 195
Configuration Examples for Configuring BFD 196
BFD Over BGP: Example 196
BFD Over OSPF: Examples 197
BFD Over Static Routes: Examples 197
BFD on Bundled VLANs: Example 197
BFD Over Bridge Group Virtual Interface: Example 198
BFD on Bundle Member Links: Examples 199
Echo Packet Source Address: Examples 202
Echo Latency Detection: Examples 202
Echo Startup Validation: Examples 203
BFD Echo Mode Disable: Examples 203
BFD Dampening: Examples 203
BFD IPv6 Checksum: Examples 204
BFD Peers on Routers Running Cisco IOS and Cisco IOS XR Software: Example 204
BFD over MPLS TE LSPs: Examples 205
BFD over MPLS TE Tunnel Head-end Configuration: Example 205

BFD over MPLS TE Tunnel Tail-end Configuration: Example 205
Where to Go Next 205
Additional References 206
Related Documents 206
Standards 206
RFCs 206
MIBs 206
Technical Assistance 207

CHAPTER 5 **Implementing EIGRP** 209
Prerequisites for Implementing EIGRP 209
Restrictions for Implementing EIGRP 210
Information About Implementing EIGRP 210
EIGRP Functional Overview 210
EIGRP Features 210
EIGRP Components 211
EIGRP Configuration Grouping 212
EIGRP Configuration Modes 212
EIGRP Interfaces 212
Redistribution for an EIGRP Process 213
Metric Weights for EIGRP Routing 213
Mismatched K Values 213
Goodbye Message 214
Percentage of Link Bandwidth Used for EIGRP Packets 214
Floating Summary Routes for an EIGRP Process 214
Split Horizon for an EIGRP Process 216
Adjustment of Hello Interval and Hold Time for an EIGRP Process 216
Stub Routing for an EIGRP Process 217
Route Policy Options for an EIGRP Process 218
EIGRP v4/v6 Authentication Using Keychain 219
How to Implement EIGRP 219
Enabling EIGRP Routing 219
Configuring Route Summarization for an EIGRP Process 221
Redistributing Routes for EIGRP 223

Creating a Route Policy and Attaching It to an EIGRP Process 224
Configuring Stub Routing for an EIGRP Process 226
Monitoring EIGRP Routing 228
Configuration Examples for Implementing EIGRP 230
Configuring a Basic EIGRP Configuration: Example 230
Configuring an EIGRP Stub Operation: Example 231
Additional References 231

CHAPTER 6 **Implementing IS-IS 233**
Prerequisites for Implementing IS-IS 233
Implementing IS-IS 233
Information About Implementing IS-IS 233
IS-IS Functional Overview 234
Key Features Supported in the Cisco IOS XR IS-IS Implementation 234
IS-IS Configuration Grouping 235
IS-IS Configuration Modes 235
Router Configuration Mode 235
Router Address Family Configuration Mode 235
Interface Configuration Mode 235
Interface Address Family Configuration Mode 235
IS-IS Interfaces 236
Multitopology Configuration 236
IPv6 Routing and Configuring IPv6 Addressing 236
Limit LSP Flooding 236
Flood Blocking on Specific Interfaces 237
Mesh Group Configuration 237
Maximum LSP Lifetime and Refresh Interval 237
Single-Topology IPv6 Support 237
Multitopology IPv6 for IS-IS 237
IS-IS Authentication 238
Nonstop Forwarding 238
ISIS NSR 240
Configuring IS-IS Adjacency Stagger 240
Multi-Instance IS-IS 241

Multiprotocol Label Switching Traffic Engineering 241
Overload Bit on Router 241
Overload Bit Configuration During Multitopology Operation 241
IS-IS Overload Bit Avoidance 242
Default Routes 242
Attached Bit on an IS-IS Instance 242
IS-IS Support for Route Tags 243
MPLS Label Distribution Protocol IGP Synchronization 243
MPLS LDP-IGP Synchronization Compatibility with LDP Graceful Restart 243
MPLS LDP-IGP Synchronization Compatibility with IGP Nonstop Forwarding 243
Label Distribution Protocol IGP Auto-configuration 244
MPLS TE Forwarding Adjacency 244
MPLS TE Interarea Tunnels 244
IP Fast Reroute 244
IS-IS Over GRE Interfaces 245
Unequal Cost Multipath Load-balancing for IS-IS 245
Enabling IS-IS and Configuring Level 1 or Level 2 Routing 246
Configuring Single Topology for IS-IS 248
Controlling LSP Flooding for IS-IS 253
Configuring Nonstop Forwarding for IS-IS 256
Configuring ISIS-NSR 258
Configuring Authentication for IS-IS 260
Configuring Keychains for IS-IS 262
Configuring MPLS Traffic Engineering for IS-IS 263
Tuning Adjacencies for IS-IS 265
Setting SPF Interval for a Single-Topology IPv4 and IPv6 Configuration 268
Customizing Routes for IS-IS 270
Configuring MPLS LDP IS-IS Synchronization 272
Tagging IS-IS Interface Routes 274
Setting the Priority for Adding Prefixes to the RIB 275
Configuring IP Fast Reroute Loop-free Alternate 277
Configure Remote Loop-Free Alternate Paths for IS-IS 278
Configuring IS-IS Overload Bit Avoidance 287
ISIS Link Group 288

Configure Link Group Profile 288
Configure Link Group Interface 291
IS-IS Cost Fallback on IOS XR Bundle-Ether Interface 292
Configuration Examples for Implementing IS-IS 294
Configuring Single-Topology IS-IS for IPv6: Example 294
Configuring Multitopology IS-IS for IPv6: Example 294
Redistributing IS-IS Routes Between Multiple Instances: Example 295
Tagging Routes: Example 295
Configuring IS-IS Overload Bit Avoidance: Example 296
Where to Go Next 296
Additional References 296

CHAPTER 7 **Implementing OSPF 299**
Prerequisites for Implementing OSPF 299
Information About Implementing OSPF 300
OSPF Functional Overview 300
Key Features Supported in the Cisco IOS XR Software OSPF Implementation 301
Comparison of Cisco IOS XR Software OSPFv3 and OSPFv2 302
OSPF Hierarchical CLI and CLI Inheritance 302
OSPF Routing Components 303
Autonomous Systems 303
Areas 303
Routers 304
OSPF Process and Router ID 305
Supported OSPF Network Types 306
Route Authentication Methods for OSPF 306
Plain Text Authentication 306
MD5 Authentication 306
Authentication Strategies 306
Key Rollover 307
Neighbors and Adjacency for OSPF 307
Designated Router (DR) for OSPF 307
Default Route for OSPF 307
Link-State Advertisement Types for OSPF Version 2 307

Link-State Advertisement Types for OSPFv3 308
Virtual Link and Transit Area for OSPF 309
Passive Interface 310
OSPF SPF Prefix Prioritization 310
Route Redistribution for OSPF 312
OSPF Shortest Path First Throttling 312
Nonstop Forwarding for OSPF Version 2 313
Graceful Shutdown for OSPFv3 313
Modes of Graceful Restart Operation 314
Graceful Restart Requirements and Restrictions 316
Warm Standby and Nonstop Routing for OSPF Version 2 317
Warm Standby for OSPF Version 3 317
Load Balancing in OSPF Version 2 and OSPFv3 317
Multi-Area Adjacency for OSPF Version 2 318
Label Distribution Protocol IGP Auto-configuration for OSPF 318
OSPF Authentication Message Digest Management 319
GTSM TTL Security Mechanism for OSPF 319
Path Computation Element for OSPFv2 319
OSPF Queue Tuning Parameters 320
OSPF IP Fast Reroute Loop Free Alternate 320
OSPF Over GRE Interfaces 320
Management Information Base (MIB) for OSPFv3 320
IGP link state 321
OSPFv2 Unequal Cost Load Balancing 321
UCMP Paths Calculation 321
Unequal Cost Multipath Load-balancing for OSPF 321
How to Implement OSPF 323
Enabling OSPF 323
Configuring Stub and Not-So-Stubby Area Types 325
Configuring Neighbors for Nonbroadcast Networks 327
Configuring Authentication at Different Hierarchical Levels for OSPF Version 2 330
Controlling the Frequency That the Same LSA Is Originated or Accepted for OSPF 334
Creating a Virtual Link with MD5 Authentication to Area 0 for OSPF 335
Examples 339

Summarizing Subnetwork LSAs on an OSPF ABR 339
Redistribute Routes into OSPF 341
Configuring OSPF Shortest Path First Throttling 344
Examples 346
Configuring Nonstop Forwarding Specific to Cisco for OSPF Version 2 346
Configuring OSPF Version 2 for MPLS Traffic Engineering 348
Examples 350
Configuring OSPFv3 Graceful Restart 352
Displaying Information About Graceful Restart 354
Enabling Nonstop Routing for OSPFv2 355
Enabling Nonstop Routing for OSPFv3 355
Configuring OSPF SPF Prefix Prioritization 356
Configuring Multi-area Adjacency 358
Configuring Label Distribution Protocol IGP Auto-configuration for OSPF 359
Configuring LDP IGP Synchronization: OSPF 360
Configuring Authentication Message Digest Management for OSPF 361
Examples 363
Configuring Generalized TTL Security Mechanism (GTSM) for OSPF 364
Examples 366
Verifying OSPF Configuration and Operation 366
Configuring OSPF Queue Tuning Parameters 368
IGP link state 369
Configuring IP Fast Reroute Loop-free Alternate 370
Enabling IPFRR LFA 370
Excluding an Interface From IP Fast Reroute Per-link Computation 371
Enabling OSPF Interaction with SRMS Server 372
Configure Remote Loop-Free Alternate Paths for OSPF 374
Configuration Examples for Implementing OSPF 382
Cisco IOS XR Software for OSPF Version 2 Configuration: Example 382
CLI Inheritance and Precedence for OSPF Version 2: Example 383
MPLS TE for OSPF Version 2: Example 384
ABR with Summarization for OSPFv3: Example 385
ABR Stub Area for OSPFv3: Example 385
ABR Totally Stub Area for OSPFv3: Example 385

Configuring OSPF SPF Prefix Prioritization: Example 385
Route Redistribution for OSPFv3: Example 387
Virtual Link Configured Through Area 1 for OSPFv3: Example 387
Virtual Link Configured with MD5 Authentication for OSPF Version 2: Example 388
OSPF Queue Tuning Parameters Configuration: Example 388
Where to Go Next 389
Additional References 389

CHAPTER 8 **Implementing IP Fast Reroute Loop-Free Alternate** 393
Prerequisites for IPv4/IPv6 Loop-Free Alternate Fast Reroute 393
Restrictions for Loop-Free Alternate Fast Reroute 393
IS-IS and IP FRR 394
Repair Paths 394
LFA Overview 395
LFA Calculation 395
Interaction Between RIB and Routing Protocols 395
Configuring Fast Reroute Support 396
Configuring IPv4 Loop-Free Alternate Fast Reroute Support: Example 398
Additional References 398

CHAPTER 9 **Implementing and Monitoring RIB** 401
Prerequisites for Implementing RIB 401
Information About RIB Configuration 402
Overview of RIB 402
RIB Data Structures in BGP and Other Protocols 402
RIB Administrative Distance 402
RIB Support for IPv4 403
RIB Statistics 403
IP Fast Reroute 404
RIB Quarantining 404
Route and Label Consistency Checker 404
How to Deploy and Monitor RIB 405
Verifying RIB Configuration Using the Routing Table 405
Disabling RIB Next-hop Dampening 406

Configuring RCC and LCC 407
Enabling RCC and LCC Background Scan 407
Configuration Examples for RIB Monitoring 408
Output of show route Command: Example 409
Output of show route backup Command: Example 409
Output of show route best-local Command: Example 409
Output of show route connected Command: Example 410
Output of show route local Command: Example 410
Output of show route longer-prefixes Command: Example 410
Output of show route next-hop Command: Example 410
Where to Go Next 411
Additional References 411

CHAPTER 10 **Implementing RIP** 413
Prerequisites for Implementing RIP 413
Information About Implementing RIP 414
RIP Functional Overview 414
Split Horizon for RIP 415
Route Timers for RIP 415
Route Redistribution for RIP 415
Default Administrative Distances for RIP 416
Routing Policy Options for RIP 417
Authentication Using Keychain in RIP 417
In-bound RIP Traffic on an Interface 418
Out-bound RIP Traffic on an Interface 418
How to Implement RIP 419
Enabling RIP 419
Customizing RIP 421
Control Routing Information 423
Creating a Route Policy for RIP 425
Configuration Examples for Implementing RIP 427
Configuring a Basic RIP Configuration: Example 427
Configuring Route Policies for RIP: Example 427
Configuring Passive Interfaces and Explicit Neighbors for RIP: Example 428

Controlling RIP Routes: Example 428
Additional References 428

CHAPTER 11 **Implementing Routing Policy** 431
Prerequisites for Implementing Routing Policy 431
Restrictions for Implementing Routing Policy 432
Information About Implementing Routing Policy 433
Routing Policy Language 433
Routing Policy Language Overview 433
Routing Policy Language Structure 434
Routing Policy Language Components 441
Routing Policy Language Usage 442
Routing Policy Configuration Basics 444
Policy Definitions 444
Parameterization 445
Parameterization at Attach Points 446
Global Parameterization 446
Semantics of Policy Application 447
Boolean Operator Precedence 447
Multiple Modifications of the Same Attribute 447
When Attributes Are Modified 448
Default Drop Disposition 449
Control Flow 449
Policy Verification 450
Policy Statements 451
Remark 451
Disposition 452
Action 454
If 454
Boolean Conditions 455
apply 456
Attach Points 457
BGP Policy Attach Points 457
OSPF Policy Attach Points 480

OSPFv3 Policy Attach Points 483
IS-IS Policy Attach Points 485
EIGRP Policy Attach Points 487
RIP Policy Attach Points 491
Attached Policy Modification 492
Nonattached Policy Modification 493
Editing Routing Policy Configuration Elements 493
Hierarchical Policy Conditions 495
Apply Condition Policies 496
Nested Wildcard Apply Policy 498
Wildcards for Route Policy Sets 499
Use Wildcards For Routing Policy Sets 499
VRF Import Policy Enhancement 503
How to Implement Routing Policy 504
Defining a Route Policy 504
Attaching a Routing Policy to a BGP Neighbor 505
Modifying a Routing Policy Using a Text Editor 507
Configuration Examples for Implementing Routing Policy 508
Routing Policy Definition: Example 508
Simple Inbound Policy: Example 508
Modular Inbound Policy: Example 509
Use Wildcards For Routing Policy Sets 510
Translating Cisco IOS Route Maps to Cisco IOS XR Routing Policy Language: Example 515
VRF Import Policy Configuration: Example 515
Additional References 515

CHAPTER 12 **Implementing Static Routes** 517
Prerequisites for Implementing Static Routes 517
Restrictions for Implementing Static Routes 517
Information About Implementing Static Routes 518
Static Route Functional Overview 518
Default Administrative Distance 518
Directly Connected Routes 518
Recursive Static Routes 519

Fully Specified Static Routes 519
Floating Static Routes 520
Configuration Examples 520
Configuring Traffic Discard: Example 520
Configuring a Fixed Default Route: Example 520
Configuring a Floating Static Route: Example 521
Where to Go Next 521
Additional References 521

CHAPTER 13 **Implementing RCMD** 523
Route Convergence Monitoring and Diagnostics 523
Configuring Route Convergence Monitoring and Diagnostics 524

Preface

The *Routing Configuration Guide for Cisco NCS 6000 Series Routers* preface contains these sections:

- Changes to This Document, on page xxi
- Communications, Services, and Additional Information, on page xxi

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

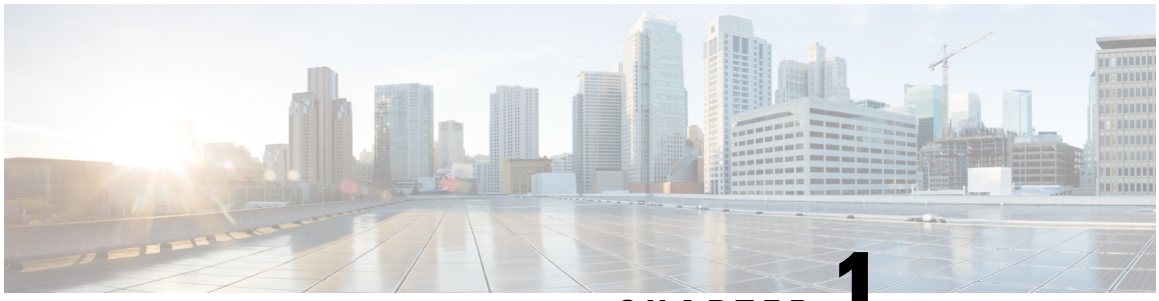
Date	Summary
July 2021	Initial release of this document.

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 NCS 6000 Series Routers*, and tells you where they are documented.

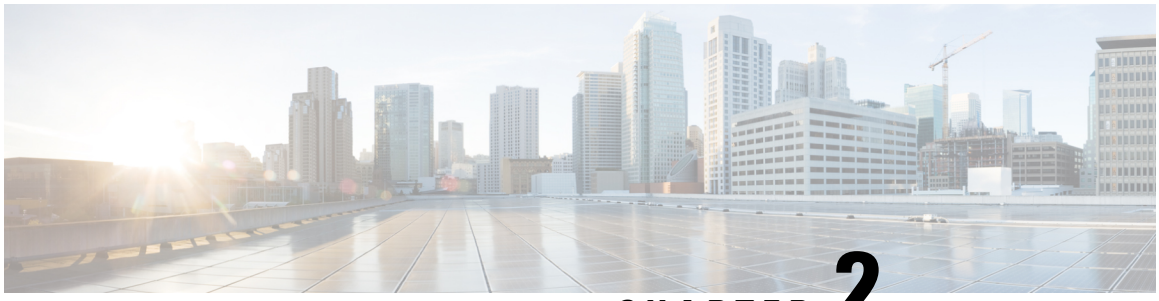
- New and Changed Routing Features, on page 1

New and Changed Routing Features

Table 2: Routing Features Added or Modified in IOS XR Release 7.4.x

Feature	Description	Changed in Release	Where Documented
None	No new features introduced	Not applicable	Not applicable

New and Changed Routing Features



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 119 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 5.0.0	This feature was introduced.
Release 5.2.3	BGP Nonstop Routing was made a default feature.

- Prerequisites for Implementing BGP, on page 4
- Information About Implementing BGP, on page 4
- Recent Prefixes Events and Trace Support, on page 56
- How to Implement BGP, on page 59
- Configuration Examples for Implementing BGP, on page 109
- Flow-tag propagation, on page 115
- Where to Go Next, on page 118
- Additional References, on page 118

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 rtfilter` command, there is no separate configuration needed to enable RTC for BGP EVPN.

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 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
  - IPv6 Unicast: 524288
  - IPv6 Labeled-unicast: 131072

  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 Enhanced Multipath Selection

Table 3: Feature History Table

Feature Name	Release Name	Description
BGP Enhanced Multipath Selection	Release 7.4.2	This feature gives you the flexibility to select unequal cost multipath (UCMP) load-balancing based on the interior gateway protocol (IGP) route metric. The IGP route metric is the sum of the metrics of all the links that belong to a path, and this feature selects the paths with lower IGP route metrics as multipath. In earlier releases, you could select BGP UCMP only based on age order, where the older path took precedence over the newer path.

The BGP multipath selection algorithm functionality enables the multipath to prefer the older paths over the new paths. Here the age order is a vital criterion for selection of the UCMP. However, this method is less optimal and is nondeterministic in terms of forwarding traffic on the network. The BGP Enhanced Multipath Selection feature allows the multipath functionality to select IGP metric.

Restrictions

- This feature is available in Internal Border Gateway Protocol.
- This feature is configurable on the following address families:
  - IPv4 Unicast
  - IPv6 Unicast
  - IPv4 Multicast
  - IPv6 Multicast
- VPNv4 does not support maximum-paths, so you cannot configure the deterministic aspect in the VPN address-family interfaces. However, you can configure the imported prefixes of VRFs with this feature.

Configuration Example

```
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# maximum-paths ibgp 2 unequal-cost deterministic
```

Running Configuration

```
router bgp 100
```

```
 address-family ipv4 unicast
  maximum-paths ibgp 2 unequal-cost deterministic
```

Verification

The following example shows you can select paths with the lower metrics as multipaths.

```
Router# show bgp ipv4 unicast 10.10.0.0/28
Paths: (128 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    22.0.1.6 (metric 20) from 198.51.100.1 (192.0.0.1)
      Origin IGP, localpref 0, valid, internal, best, group-best, multipath
      Received Path ID 1, Local Path ID 1, version 12611
      Originator: 192.0.0.1, Cluster list: 198.51.100.1
…

Path #64: Received by speaker 0
  Not advertised to any peer
  Local
    23.0.11.6 (metric 30) from 203.0.113.1 (210.0.0.10)
      Origin IGP, localpref 0, valid, internal, multipath
      Received Path ID 32, Local Path ID 0, version 0
      Originator: 210.0.0.10, Cluster list: 203.0.113.1, 200.0.0.10

Path #65: Received by speaker 0
  Not advertised to any peer
  Local
    24.0.1.6 (metric 40) from 192.0.2.254 (211.0.0.0)
      Origin IGP, localpref 0, valid, internal
      Received Path ID 1, Local Path ID 0, version 0
      Originator: 211.0.0.0, Cluster list: 192.0.2.254, 201.0.0.0, 202.0.0.0
…

  Path #128: Received by speaker 0
  Not advertised to any peer
  Local
    25.0.23.6 (metric 50) from 198.51.100.233 (195.0.0.23)
      Origin IGP, localpref 0, valid, internal
      Received Path ID 32, Local Path ID 0, version 0
      Originator: 195.0.0.23, Cluster list: 198.51.99.255
```

The following example displays the BGP multipaths installed in the RIB.

```
Router# show route ipv4 200.0.0.0/28
Routing entry for 200.0.0.0/28
 Known via "bgp 1", distance 200, metric 0, type internal
 Installed Oct 17 04:06:41.027 for 00:01:22
 outing Descriptor Blocks
    10.0.1.6, from 198.51.100.1, BGP multi path
      Route metric is 0
    10.0.2.6, from 198.51.100.1, BGP multi path
      Route metric is 0
    …
    10.0.32.6, from 198.51.100.1, BGP multi path
      Route metric is 0
    198.51.100.253, from 203.0.113.1, BGP multi path
      Route metric is 0
    …
    198.51.100.252, from 203.0.113.1, BGP multi path
      Route metric is 0
  No advertising protos.
```

The following example displays the BGP multipaths installed in Cisco Express Forwarding.

```
Router# show cef ipv4 200.0.0.0/28 detail
Level 1 - Load distribution: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
 54 55 56 57 58 59 60 61 62 63
 [0] via 10.0.1.6/32, recursive
 [1] via 10.0.2.6/32, recursive

 [62] via 203.0.113.211/32, recursive
 [63] via 203.0.112.211/32, recursive

  via 10.0.1.6/32, 257 dependencies, recursive, bgp-multipath [flags 0x6080]
  path-idx 0 NHID 0x0 [0x7a6cdf90 0x0]
  next hop 22.0.1.6/32 via 22.0.0.0/8

  Load distribution: 0 (refcount 1)
  Hash  OK  Interface              Address
  0     Y   TenGigE0/1/0/0/7       remote

 via 203.0.113.211/32, 257 dependencies, recursive, bgp-multipath [flags 0x6080]
 path-idx 62 NHID 0x0 [0x7a6ce058 0x0]
  next hop 203.0.112.211/32 via 203.0.0.0/8

  Load distribution: 0 (refcount 1)

  Hash  OK  Interface                 Address
  2     Y   TenGigE0/1/0/0/4          remote

  via 203.0.32.6/32, 257 dependencies, recursive, bgp-multipath [flags 0x6080]
  path-idx 63 NHID 0x0 [0x7a6ce058 0x0]
  next hop 203.0.32.6/32 via 203.0.0.0/8

  Load distribution: 0 (refcount 1)

  Hash  OK  Interface                 Address
  63    Y   TenGigE0/1/0/0/4          remote
```

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 recursed metric events trigger a best-path recalculation.

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

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

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

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

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

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

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 Table Walk

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

Reordered Address Family Processing

The 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 labeled unicast
- IPv4 unicast
- IPv6 unicast
-

New Thread for Next-Hop Processing

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

show, clear, and debug Commands

The **show bgp nexthops** command provides statistical information about next-hop notifications, the amount of time spent in processing those notifications, and details about each next hop registered with the RIB. The **clear bgp nexthop performance-statistics** command ensures that the cumulative statistics associated with the processing part of the next-hop **show** command can be cleared to help in monitoring. The **clear bgp nexthop registration** command performs an asynchronous registration of the next hop with the RIB. See the *BGP Commands on Cisco IOS XR Software* module of *Routing Command Reference for Cisco NCS 6000 Series 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 .

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.

```
RP/0/RP0/CPU0:router(config)# as-format [asdot | asplain]
RP/0/RP0/CPU0:router(config)# as-format asdot
```

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

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)#
```

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
  ```

Configuration Templates

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

The **af-group** command is used to group address family-specific neighbor commands within an IPv4, IPv6, address family. Neighbors that have the same address family configuration are able to use the address family group (af-group) name for their address family-specific configuration. A neighbor inherits the configuration from an address family group by way of the **use** command. If a neighbor is configured to use an address family group, the neighbor (by default) inherits the entire configuration from the address family group. However, a neighbor does not inherit all of the configuration from the address family group if items are explicitly configured for the neighbor. The address family group configuration is entered under the BGP router configuration mode. The following example shows how to enter address family group configuration mode :

```
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 **show bgp neighbor** output shows the cumulative number for the *Prefix advertised* count if the same prefixes are withdrawn and re-advertised.

   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
```

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 XR EXEC 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:routerconfig-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
```

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 NCS 6000 Series 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 NCS 6000 Series 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 25
BGP Update Group , on page 25

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 inInfluencing Route Preference in a Multiexit IGP Network, on page 27.

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

SeeBGP Best Path Algorithm, on page 28 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

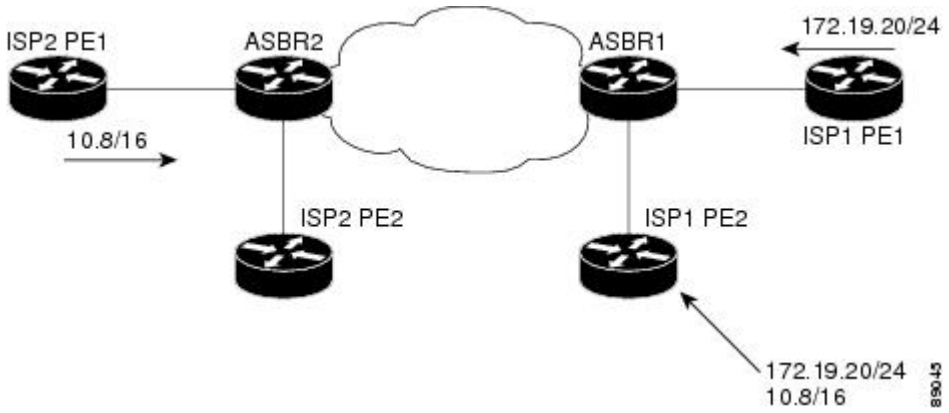
The single advertised path includes the aggregate cost communities as follows:

{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.

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 28 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_58 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_58.
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_58.
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_58.
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_58, 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 NCS 6000 Series 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 4: BGP Default Administrative Distances, on page 32.

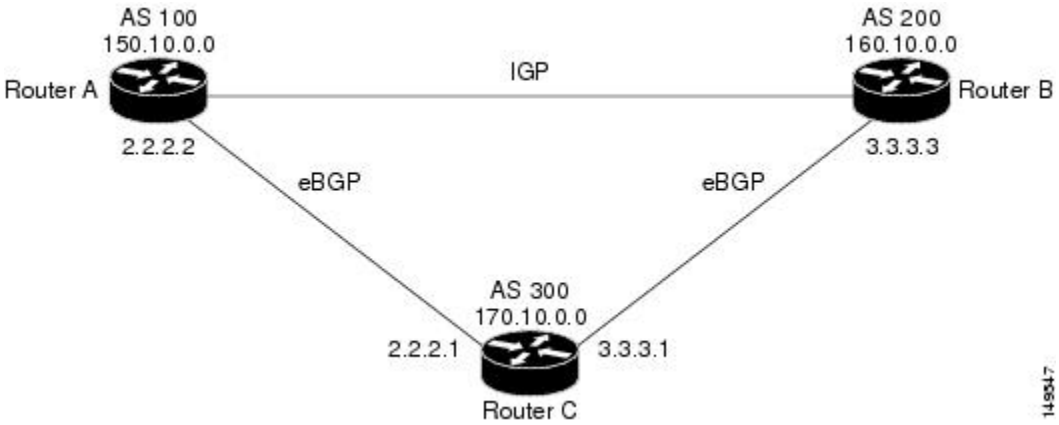
Table 4: BGP Default Administrative Distances

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 2: Back Door Example , on page 33.

Figure 2: Back Door Example



In Figure 2: Back Door Example , on page 33, 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.

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

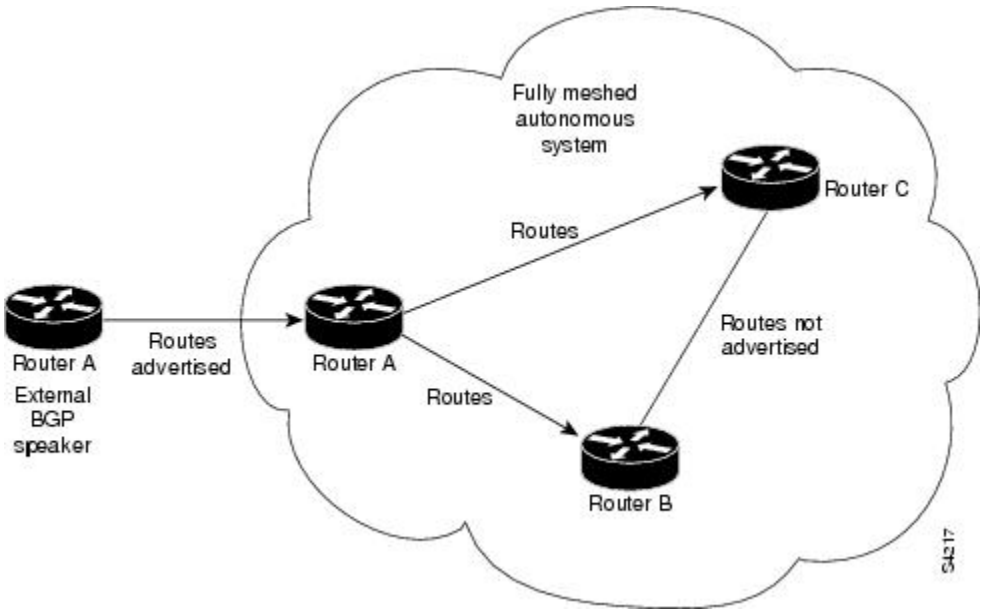
One way to reduce the iBGP mesh is to divide an autonomous system into multiple subautonomous systems and group them into a single confederation. To the outside world, the confederation looks like a single autonomous system. Each autonomous system is fully meshed within itself and has a few connections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

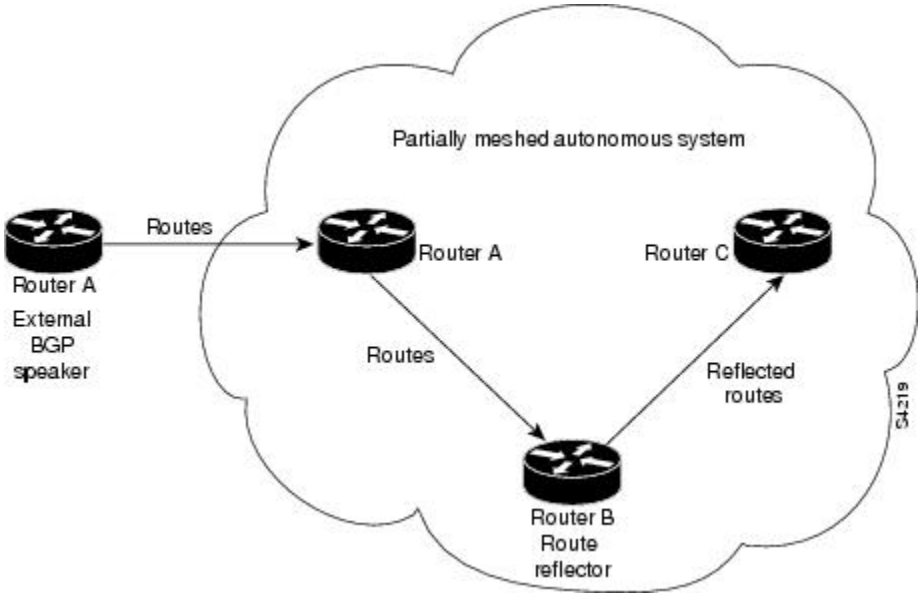
Figure 3: Three Fully Meshed iBGP Speakers, on page 35 illustrates a simple iBGP configuration with three iBGP speakers (routers A, B, and C). Without route reflectors, when Router A receives a route from an external neighbor, it must advertise it to both routers B and C. Routers B and C do not readvertise 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 3: 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 4: Simple BGP Model with a Route Reflector, on page 35 , 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 4: 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 5: More Complex BGP Route Reflector Model

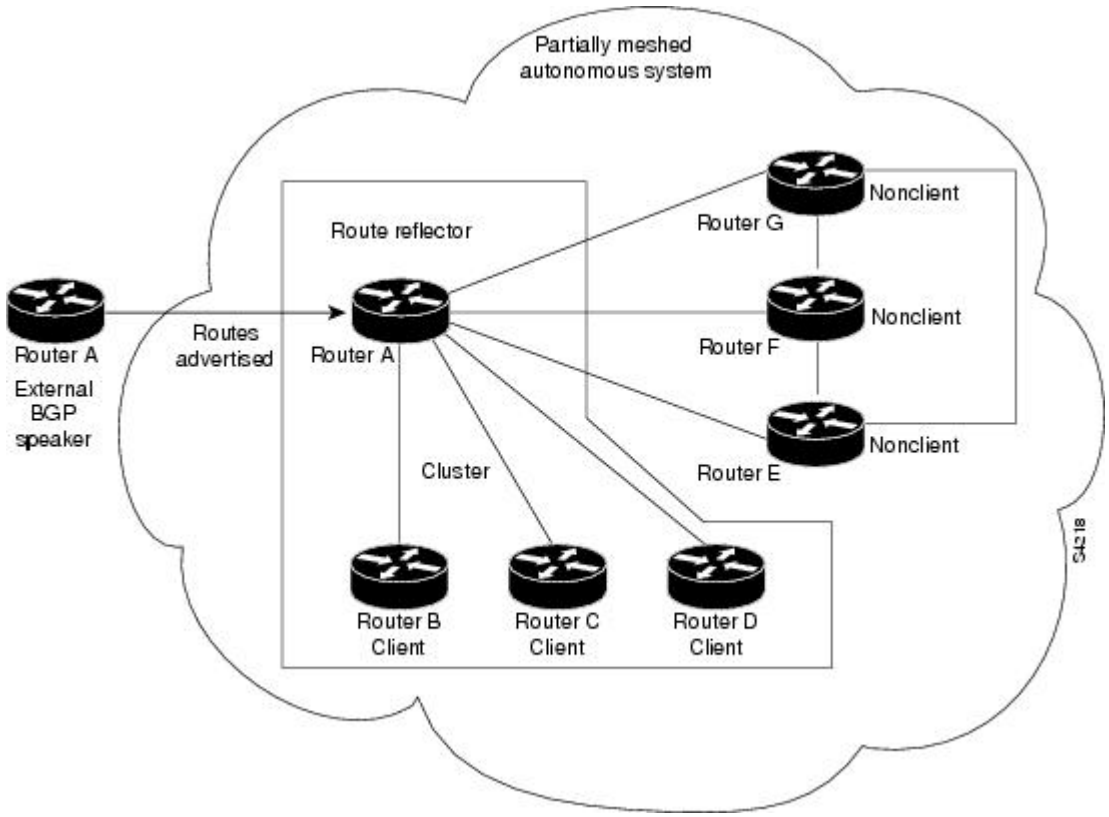


Figure 5: More Complex BGP Route Reflector Model, on page 36 illustrates a more complex route reflector scheme. Router A is the route reflector in a cluster with routers B, C, and D. Routers E, F, and G are fully meshed, nonclient routers.

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.

Remotely Triggered Null Route Filtering with RPL Next-hop Discard Configuration

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

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

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

Note RTBH is not supported in cases such as L3VPN iBGP route over NULL0.

Note On Jericho2 TCAM-based platforms, when you configure a NULL interface, both destination-based RTBH filtering (D-RTBH) and source-based RTBH filtering (S-RTBH) are triggered.

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.

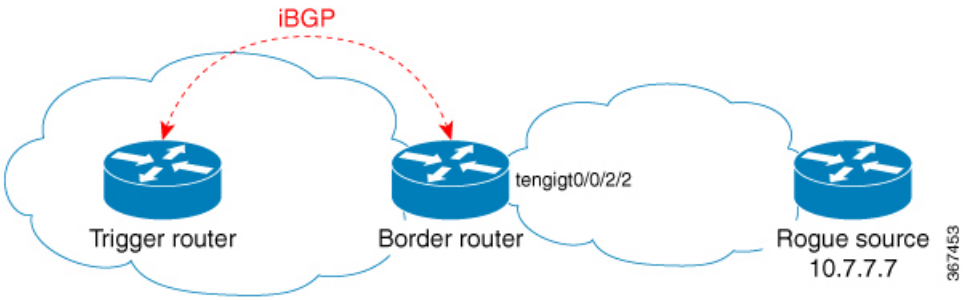
RTBH filtering sets the next-hop of the victim's prefix to the null interface. The traffic destined to the victim is dropped at the ingress.

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 6: 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**

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.

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 NCS 6000 Series 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 Session Authentication and Integrity using TCP Authentication Option Overview

BGP Session Authentication and Integrity using TCP Authentication Option feature enables you to use stronger Message Authentication Codes that protect against replays, even for long-lived TCP connections. This feature also provides more details on the association of security with TCP connections than TCP MD5 Signature option (TCP MD5).

This feature supports the following functionalities of TCP MD5:

- Protection of long-lived connections such as BGP and LDP.
- Support for larger set of MACs with minimal changes to the system and operations

BGP Session Authentication and Integrity using TCP Authentication Option feature supports IPv6. It supports these two cryptographic algorithms: HMAC-SHA-1-96 and AES-128-CMAC-96.

You can use two sets of keys, namely Master Key Tuples and traffic keys to authenticate incoming and outgoing segments.

This feature applies different option identifier than TCP MD5. This feature cannot be used simultaneously with TCP MD5.

Master Key Tuples

Traffic keys are the keying material used to compute the message authentication codes of individual TCP segments.

The BGP Session Authentication and Integrity using TCP Authentication Option (AO) feature uses the existing keychain fucntionality to define the key string, message authentication codes algorithm, and key lifetimes.

Master Key Tuples (MKTs) enable you to derive unique traffic keys, and to include the keying material required to generate those traffic keys. MKTs indicate the parameters under which the traffic keys are configured. The parameters include whether TCP options are authenticated, and indicators of the algorithms used for traffic key derivation and MAC calculation.

Each MKT has two identifiers, namely **SendID** and a **RecvID**. The SendID identifier is inserted as the KeyID identifier of the TCP AO option of the outgoing segments.The **RecvID** is matched against the TCP AO KeyID of the incoming segments.

Configure BGP Session Authentication and Integrity using TCP Authentication Option

This section describes how you can configure BGP Session Authentication and Integrity using TCP Authentication Option (TCP AO) feature :

- Configure Keychain

Note Configure send-life and accept-lifetime keywords with identical values in the keychain configuration, otherwise the values become invalid.

- Configure TCP

Note The Send ID and Receive ID you configured on the device must match the Receive ID and Send ID configured on the peer respectively.

- Configure BGP

Configuration Example

Configure a keychain.

```
Router# configure
Router#(config)# key chain tcpao1
Router#(config-tcpao1)# key 1
Router#(config-tcpao1-1)# cryptographic-algorithm HMAC-SHA-1-96
Router#(config-tcpao1-1)# key-string keys1
Router#(config-tcpao1-1)# send-lifetime 16:00:00 march 3 2018 infinite
Router#(config-tcpao1-1)# accept-lifetime 16:00:00 march 3 2018 infinite
```

Configure TCP

```
Router# tcp ao
Router(config-tcp-ao)# keychain tcpao1
Router(config-tcp-ao-tpcao1)# key 1 sendID 5 receiveID 5
/* Configure BGP */
Router#(config-bgp)# router bgp 1
Router(config-bgp)# bgp router-id 10.101.101.1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.51.51.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# ao tcpao1 include-tcp-options disable accept-ao-mismatch-connection
```

Configure BGP

```
Router#(config-bgp)# router bgp 1
Router(config-bgp)# bgp router-id 10.101.101.1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.51.51.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# ao tcpao1 include-tcp-options disable accept-ao-mismatch-connection
```

Verification

Verify the keychain information configured for BGP Session Authentication and Integrity using TCP Authentication Option feature.

```
Router# show bgp sessions | i 10.51.51.1
Wed Mar 21 12:55:57.812 UTC
10.51.51.1  default  1     1     0     0    Established  None
```

The following output displays details of a key, such as Send Id, Receive Id, and cryptographic algorithm.

```
Router# show bgp sessions | i 10.51.51.1
Wed Mar 21 12:55:57.812 UTC
10.51.51.1  default  1     1     0     0    Established  None
```

The following output displays the state of the BGP neighbors.

```
Router# show bgp sessions | i 10.51.51.1
Wed Mar 21 12:55:57.812 UTC
10.51.51.1  default  1     1     0     0    Established  None
```

The following output displays the state of a particular BGP neighbor.

```
Router# show bgp sessions | i 10.51.51.1
Wed Mar 21 12:55:57.812 UTC
10.51.51.1  default  1     1     0     0    Established  None
```

The following output displays brief information of the protocol control block (PCB) of the neighbor.

```
Router# show tcp brief | i 10.51.51.2
Wed Mar 21 12:55:13.652 UTC
0x143df858 0x60000000  0  0  10.51.51.2:43387  10.51.51.1:179 ESTAB
```

The following output displays authentication details of the PCB:

```
Router# show tcp detail pcb 0x143df858 location 0/rsp0/CPU0 | begin Authen
Wed Mar 21 12:56:46.129 UTC
Authentication peer details:
    Peer: 10.51.51.1/32, OBJ_ID: 0x40002fd8
    Port: BGP, vrf_id: 0x60000000, type: AO, debug_on:0
    Keychain_name: tcpao1, options: 0x00000000, linked peer: 0x143e00 □ Keychain name
    Send_SNE: 0, Receive_SNE: 0, Send_SNE_flag: 0
    Recv_SNE_flag: 0, Prev_send_seq: 4120835405, Prev_receive_seq: 2461932863
    ISS: 4120797604, IRS: 2461857361
    Current key: 2
    Traffic keys: send_non_SYN: 006a2975, recv_non_SYN: 00000000
    RNext key: 2
    Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
    Last 1 keys used:
        key: 2, time: Mar 20 03:52:35.969.151, reason: No current key set
```

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.

- 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.

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 lable 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 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

TheBorder 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.

Persistent Loadbalancing

Traditional ECMP or equal cost multipath loadbalances traffic over a number of available paths towards a destination. When one path fails, the traffic gets re-shuffled over the available number of paths. This flow distribution can be a problem in data center loadbalancing.

Persistent Loadbalancing or Sticky ECMP defines a prefix in such a way that it do not rehash flows on existing paths and only replace those bucket assignments of the failed server. The advantage is that the established sessions to servers will not get rehashed.

The following section describes how you can configure persistent load balancing:

```
/*Configure persistent load balancing. */

Router(config)# router bgp 7500
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy sticky-ecmp
Router(config-bgp-af)# bgp attribute-download
Router(config-bgp-af)# maximum-paths ebgp 64
Router(config-bgp-af)# maximum-paths ibgp 32
Router(config-bgp-af)# exit
Router(config-bgp)# exit
Router(config)# route-policy sticky-ecmp
Router(config-rpl)# if destination in (192.1.1.1/24) then
Router(config-rpl-if)# set load-balance ecmp-consistent
Router(config-rpl-if)# else
Router(config-rpl-else)# pass
Router(config-rpl-else)# endif
RP/0/0/CPU0:ios(config-rpl)# end-policy
RP/0/0/CPU0:ios(config)#

/* Enable autocovery and hence recover the original hashing state
after failed paths become active. */
Router(config)# cef consistent-hashing auto-recovery

/* Recover to the original hashing state after failed paths come up
and avoid affecting newly formed flows after path failure.  */
Router(config)# clear route 192.0.2.0/24
```

Running Configuration

```
/* Configure persistent loadbalancing. */
router bgp 7500
 address-family ipv4 unicast
  table-policy sticky-ecmp
  bgp attribute-download
  maximum-paths ebgp 64
  maximum-paths ibgp 32


cef consistent-hashing auto-recovery
```

```
clear route 192.0.2.0/24
```

Verification

Verify that the path distribution with persistent loadbalancing is configured.

The following show output displays the status of path distribution before a link fails. In this output, three paths are identified with three next hops (10.1/2/3.0.1) through three different GigabitEthernet interfaces.

```
show cef 192.0.2.0/24
 LDI Update time Sep  5 11:22:38.201
   via 10.1.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 0 NHID 0x0 [0x57ac4e74 0x0]
    next hop 10.1.0.1/32 via 10.1.0.1/32
   via 10.2.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 1 NHID 0x0 [0x57ac4a74 0x0]
    next hop 10.2.0.1/32 via 10.2.0.1/32
   via 10.3.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 2 NHID 0x0 [0x57ac4f74 0x0]
    next hop 10.3.0.1/32 via 10.3.0.1/32


    Load distribution (consistent): 0 1 2 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   GigabitEthernet0/0/0/0    10.1.0.1
    1     Y   GigabitEthernet0/0/0/1    10.2.0.1
    2     Y   GigabitEthernet0/0/0/2    10.3.0.1
```

The following show output displays the status of the path distribution after a link fails. The replacement of bucket 1 with GigabitEthernet 0/0/0/0 and the "*" symbol denotes that this path is a replacement for a failed path.

```
show cef 192.0.2.0/24
 LDI Update time Sep  5 11:23:13.434
   via 10.1.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 0 NHID 0x0 [0x57ac4e74 0x0]
    next hop 10.1.0.1/32 via 10.1.0.1/32
   via 10.3.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 1 NHID 0x0 [0x57ac4f74 0x0]
    next hop 10.3.0.1/32 via 10.3.0.1/32

    Load distribution (consistent) : 0 1 2 (refcount 1)
    Hash  OK  Interface                 Address
    0     Y   GigabitEthernet0/0/0/0    10.1.0.1
 1*    Y   GigabitEthernet0/0/0/0    10.1.0.1
    2     Y   GigabitEthernet0/0/0/2    10.3.0.1
```

BGP Selective Multipath

Traditional BGP multipath feature allows a router receiving parallel paths to the same destination to install the multiple paths in the routing table. By default, this multipath feature is applied to all configured peers. BGP selective multipath allows application of the multipath feature only to selected peers.

The BGP router receiving multiple paths is configured with the **maximum-paths ... selective** option. The iBGP/eBGP neighbors sharing multiple paths are configured with the **multipath** option, while being added as neighbors on the BGP router.

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

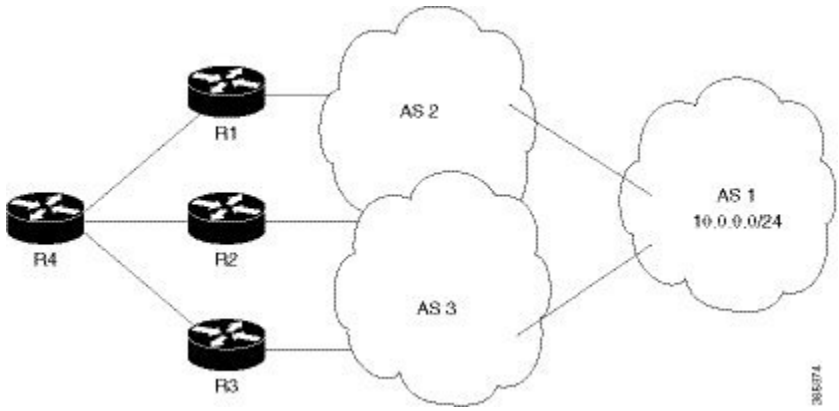
- BGP selective multipath does not impact best path calculations. A best path is always included in the set of multipaths.
- For VPN prefixes, the PE paths are *always* eligible to be multipaths.

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 7: 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.

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 NCS 6000 Series Routers* and *Interface and Hardware Component Command Reference for the Cisco NCS 6000 Series 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.

- 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.

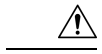
BGP Prefix Independent Convergence for RIB and FIB

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 traffic outage in the network. As a workaround, you must route the traffic to the backup path manually before configuring the **neighbor shutdown** command.

Note For releases prior to Cisco IOS XR release 7.3.x, a timer is implemented and enabled to force using backup path after convergence for 4 minutes to prevent the traffic loss for BGP PIC Edge scenario with dual-homed CE and BFD/BGP running between PE and CE routers. This timer however can cause side effect resulting traffic outage under certain scenarios. You can use the **cef fast-reroute follow bgp-pic** command to disable the timer. Starting IOS XR release 7.3.x, this command has been deprecated and the default system behavior is to disable the 4-minute timer.

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.

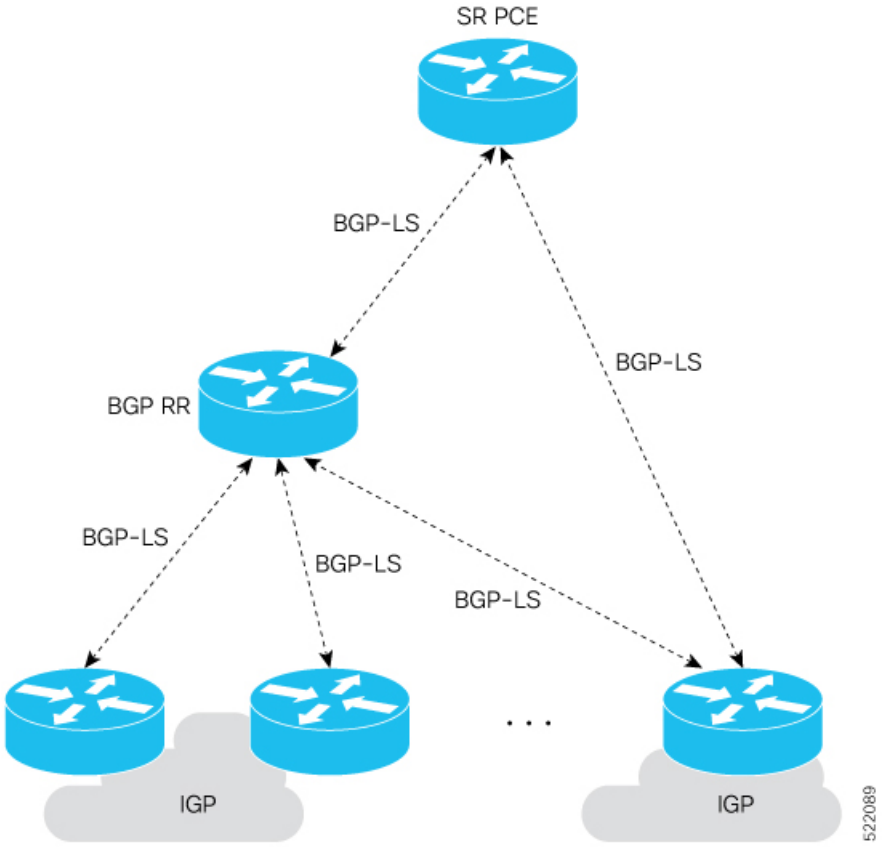
BGP Link-State

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

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

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

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



Exchange Link State Information with BGP Neighbor

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

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

IGP Link-State Database Distribution

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

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

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

Usage Guidelines and Limitations

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

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.

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.

Note With dynamic route-leaking enabled, BGP bestpath change suppression for eBGP paths might be skipped. BGP convergence might be impacted.

Recent Prefixes Events and Trace Support

The Recent Prefixes Events and Trace Support feature enables you to obtain per prefix level churning information without the use of debug commands. The show commands associated with this feature provide you a recent history of major events at the prefix level. They display the last eight events for the last 100 churning number of prefixes across an address family.

The following address families support this feature:

- IPv4 Unicast
- IPv6 Unicast
- IPv4 Multicast
- IPv6 Multicast
- VPNv4 Unicast
- VPNv6 Unicast
- BGP Link-State
- L2VPN EVPN
- IPv4 FlowSpec

Retrictions

The following restrictions apply to recent prefixes only. They do not apply to trace support.

- You can only track remote prefixes and path updates. You cannot track internal event trigger or local prefixes updates.
- You cannot track the events when the neighbor session goes down

Verification

Use the following command to check the events for a specific prefix.

```
Router# show bgp ipv4 unicast recent-prefixes 192.168.112.0/24 priv$

P/0/RP0/CPU0:root#
Tue Jan 21 10:30:44.488 UTC

Address-Family: IPv4 Unicast Route-Distinguisher: 0:0:0
192.168.112.0/24
Event History [Total events: 8]
------------------------------------
Time                 Event     Context1  Context2  Context3
====                 =====     =====     =====     =====
```

```
Dec 19 16:39:53.329  Withdraw  0x3010101  0x0     0x4000000000020004
Dec 19 16:39:53.330  Create    0x3010101  0x0     0x4000000000020005
Dec 19 16:39:53.330  Modify    0x3010101  0x0     0x4000000000020005
Dec 19 16:40:42.717  Create    0x3010101  0x0     0x4000000000020005
Dec 19 18:16:33.318  Create    0x3010101  0x0     0x4000000000020005
Jan 2 13:36:18.595  Modify     0x3010101  0x0     0x4000000000020005
Jan 2 15:16:00.344  Duplicate  0x3010101  0x0     0x4000000000020005
Jan 14 15:56:28.561 Duplicate 0x3010101   0x0     0x4000000000020005
------------------------------------
```

Verify the route distinguishers and corresponding prefix.

```
Router# show bgp l2vpn recent-prefixes

Address-Family     Route-Distinguisher         Prefix
==============     ===================         ==============
L2VPN EVPN         0:0:0 [5][0][32]           [198.51.100.22]/24
L2VPN EVPN         10.5.0.1:100 [5][0][32]    [192.0.2.1]/24
L2VPN EVPN         10.5.0.1:100 [5][0][32]    [192.0.2.2]/24
L2VPN EVPN         10.5.0.1:100 [5][0][32]    [192.0.2.3]/24
L2VPN EVPN         10.5.0.1:100 [5][0][32]    [192.0.2.4]/24
```

Verify recently updated or deleted prefixes.

```
Router# show bgp ipv4 unicast recent-prefixes

Address-Family   Route-Distinguisher Prefix
================ =================== ==============
IPv4 Unicast     0:0:0               10.1.1.1/32
IPv4 Unicast     0:0:0               10.1.1.101/32
IPv4 Unicast     0:0:0               10.1.1.100/32
IPv4 Unicast     0:0:0               10.1.1.99/32
IPv4 Unicast     0:0:0               10.1.1.98/32
IPv4 Unicast     0:0:0               10.1.1.93/32
```

Verify recently updated or deleted prefixes with timestamps and related contexts.

```
Router# show bgp ipv4 unicast recent-prefixes private

Address-Family: IPv4 Unicast Route-Distinguisher: 0:0:0
10.1.1.10/32
Event History [Total events: 4]
------------------------------------
Time                Event     Context1    Context2  Context3
====                =====     ========    ========  ========
Jul 24 17:03:58.357 Create    0x13000001  0x0       0x4000000000000007
Jul 24 17:04:12.365 Withdraw  0x13000001  0x0       0x4000000001040006
Jul 24 17:04:31.625 Create    0x13000001  0x0       0x4000000000000007
Jul 24 17:04:39.880 Duplicate 0x13000001 0x0       0x4000000000000007
```

Verify recent history of major events in the link-state database of a network advertised through BGP.

```
Router# show bgp link-state link-state recent-prefixes

Address-Family: Link-state Link-state Route-Distinguisher: 0:0:0
[E][B][I0x0][N[c1][b19.0.0.1][q19.0.0.1]][R[c200][q19.0.0.2]][L[i26.0.101.100][n29.0.1.30]]/600
Event History [Total events: 4]
------------------------------------
Time Event Context1 Context2 Context3
==== ===== ===== ===== =====
Aug 1 15:45:25.171 Create 0x13000001 0x0 0x4000000000020005
```

Reasons for not Advertising BGP Prefix to a Peer

The following are the categories of reasons for which a BGP prefix may not be advertised to a peer or a set of peers. The exact reason for which the BGP prefix is not advertised is displayed in the output of the show bgp ipv4 unicast update-group performance-statistics command.

- Path element not applicable
- Path not available
- Block stitching route targer (RT) constraint
- Block RT constraint network layer reachability information (NLRI)
- Imported path to non-customer edge (CE) neighbor
- VPN only path to CE neighbor
- External peer with no export
- Encapsulation mismatch (VxLAN)
- Sender Autonomous System (AS)
- Non-client to non-client
- Cluster identifier not set
- Client to non-client for cluster
- No PIM feedback for eBGP neighbor
- No PIM feedback
- PIM withdraw Feedback
- Wait for PIM feedback
- Prefix-based outbound route filter (ORF)
- RT type mismatch
- No out-policy for eBGP neighbor
- Out-policy
- Nexthop and label select fail
- V6 nexthop for V4 NLRI non-extended encoding capable
- No label
- Net suppressed
- No second label
- Dropped by RT filter
- Dropped by MVPN neighbor filter
- Oversized
- Split horizon update

Verification

The below example shows how to display performance statisticsfor a unadvertized prefix without enabling debug commands and checking the logs.

BGP prefix may not be advertized to a peer or a set of peers. The below example shows how to display the total numbers of prefixes not advertising in any AFI or SAFI, including repeating counts on 1 or more prefixes

```
Router# show bgp update-group performance-statistics

Update group for IPv4 Unicast, index 0.1:
..
Update timer last processed: Sep 23 00:10:15.350
  Not-Advertised Stats:
    Non-Client to Non-Client       : 105        Sep 23 00:10:15.350
    Path Not Available       : 132        Sep 23 00:10:15.350
```

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 neighborship between two peers, BGP checks if the two peers are directly connected. If the peers are not directly connected, BGP does not try to establish a relationship by default. If two BGP peers are not directly connected and peering is required between the loop backs of the routers, you can use the **ignore-connected-check** command. This command overrides the default check that BGP performs which is to verify if source IP in BGP control packets is in same network as that of destination. In this scenario, a TTL value of 1 is sufficient if **ignore-connected-check** is used.

Configuring **egp-multihop** *ttl* is needed when the peers are not directly connected and there are more routers in between. If the **egp-multihop** *ttl* command is not configured, eBGP sets the TTL of packets carrying BGP messages to 1 by default. When eBGP needs to be setup between routers which are more than one hop away, you need to configure a TTL value which is at least equal to the number of hops between them. For example, if there are 2 hops (R2, R3) between two BGP peering routers R1 and R4, you need to set a TTL value of 3.

SUMMARY STEPS

1. **configure**
2. **route-policy** *route-policy-name*
3. **end-policy**
4. Use the **commit** or **end** command.
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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

Step 2	**route-policy** *route-policy-name*
	Example:
	`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`
		(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-rpl)# else RP/0/RP0/CPU0:router(config-rpl)# pass RP/0/RP0/CPU0:router(config-rpl)# endif	
Step 3	**end-policy** **Example:** RP/0/RP0/CPU0:router(config-rpl)# end-policy	(Optional) Ends the definition of a route policy and exits route policy configuration mode.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**configure** **Example:** RP/0/RP0/CPU0:router# configure	Enters XR Config mode.
Step 6	**router bgp** *as-number* **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 7	**bgp router-id** *ip-address* **Example:** RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 192.168.70.24	Configures the local router with a specified router ID.
Step 8	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** 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 9	**exit** **Example:** RP/0/RP0/CPU0:router(config-bgp-af)# exit	Exits the current configuration mode.

	Command or Action	Purpose
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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Multiple BGP Instances for a Specific Autonomous System

Perform this task to configure multiple BGP instances for a specific autonomous system.

All configuration changes for a single BGP instance can be committed together. However, configuration changes for multiple instances cannot be committed together.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number* [**instance** *instance name*]
3. **bgp router-id** *ip-address*

4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* [**instance** *instance name*] **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** *ip-address* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**bgp confederation identifier** *as-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp confederation identifier 5`	Specifies a BGP confederation identifier.

Step 4	**bgp confederation peers** *as-number*
	Example:
	`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`
		Specifies that the BGP autonomous systems belong to a specified BGP confederation identifier. You can associate multiple AS numbers to the same confederation identifier, as shown in the example.

Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **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** *keepalive hold-time* **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** *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 5	**timers** *keepalive hold-time* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**bgp default local-preference** *value* **Example:** `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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**default-metric** *value* **Example:** `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).
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `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** *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 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** *weight-value* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# weight 41150`	Assigns a weight to all routes learned through the neighbor.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 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:** `RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath med always`	Configures the BGP speaker in the specified autonomous system to compare MEDs among all the paths for the prefix, regardless of the autonomous system from which the paths are received.

	Command or Action	Purpose
Step 5	**bgp bestpath med confed** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath med confed`	Enables BGP software to compare MED values for paths learned from confederation peers.
Step 6	**bgp bestpath as-path ignore** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath as-path ignore`	Configures the BGP software to ignore the autonomous system length when performing best-path selection.
Step 7	**bgp bestpath compare-routerid** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath compare-routerid`	Configure the BGP speaker in the autonomous system to compare the router IDs of similar paths.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 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	**network** { *ip-address* / *prefix-length* | *ip-address mask* } **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 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** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# aggregate-address 10.0.0.0/8 as-set`	Creates an aggregate address. The path advertised for this route is an autonomous system set consisting of all elements contained in all paths that are being summarized. • The **as-set** keyword generates autonomous system set path information and community information from contributing paths. • The **as-confed-set** keyword generates autonomous system confederation set path information from contributing paths. • The **summary-only** keyword filters all more specific routes from updates. • The **route-policy** *route-policy-name* keyword and argument specify the route policy used to set the attributes of the aggregate route.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.

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 IGPs 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**bgp redistribute-internal** **Example:** `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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters XR Config mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# configure`	
Step 2	**router bgp** *as-number* **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: • **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*] **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP Route Dampening

Perform this task to configure and monitor BGP route dampening.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*]
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 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	**bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# bgp dampening 30 1500 10000 120`	Configures BGP dampening for the specified address family. • *half-life*—(Optional) Time (in minutes) after which a penalty is decreased. Once the route has been assigned a penalty, the penalty is decreased by half after the half-life period (which is 15 minutes by default). Penalty reduction happens every 5 seconds. Range of the half-life period is from 1 to 45 minutes. • *reuse*—(Optional) Value for route reuse if the flapping route penalty decreases and falls below the reuse value. When this happens, the route is unsuppressed. The process of unsuppressing routes occurs at 10-second increments. Range is 1 to 20000. • *suppress*—(Optional) Maximum penalty value. Suppress a route when its penalty exceeds the value specified. When this happens, the route is suppressed. Range is 1 to 20000.

	Command or Action	Purpose
		• *max-suppress-time*—(Optional) Maximum time (in minutes) a route can be suppressed. Range is 1 to 255. If the *half-life* value is allowed to default, the maximum suppress time defaults to 60 minutes. • **route-policy** *route-policy-name* —(Optional) Specifies the route policy to use to set dampening parameters.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Applying 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* module of *Routing Configuration Guide for Cisco NCS 6000 Series 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **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** *policy-name* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 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	**distance bgp** *external-distance internal-distance local-distance* **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# distance bgp 20 20 200`	Sets the external, internal, and local administrative distances to prefer one class of routes over another. The higher the value, the lower the trust rating.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 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 (?).

	Command or Action	Purpose
Step 4	exit **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# exit`	Exits the current configuration mode.
Step 5	**neighbor-group** *name* **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** *as-number* **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** *route-policy-name* { **in** | **out** } **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.
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:**	(Optional) Specifies that the BGP neighbor inherit configuration from the specified neighbor group.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group nbr-grp-A`	
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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**bgp cluster-id** *cluster-id* **Example:** `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** *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 5	**remote-as** *as-number* **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** { **ipv4** | **ipv6** } **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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* module of *Cisco 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

Step 2	**route-policy** *name*
	Example:
	`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`
		(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.

	Command or Action	Purpose
Step 3	**end-policy** **Example:** `RP/0/RP0/CPU0:router(config-rpl)# end-policy`	(Optional) Ends the definition of a route policy and exits route policy configuration mode.
Step 4	**router bgp** *as-number* **Example:** `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 5	**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 6	**address-family** { **ipv4** | **ipv6** } **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** *route-policy-name* { **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP 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** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**attribute-filter group** *attribute-filter group name* **Example:** `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.
Step 4	**attribute** *attribute code* { **discard** | **treat-as-withdraw** } **Example:** `RP/0/RP0/CPU0:router(config-bgp-attrfg)# attribute 24 discard`	Specifies a single or a range of attribute codes and an associated action. The allowed actions are: • Treat-as-withdraw— Considers the update message for withdrawal. The associated IPv4-unicast or MP_REACH NLRIs, if present, are withdrawn from the neighbor's Adj-RIB-In. • Discard Attribute— Discards this attribute. The matching attributes alone are discarded and the rest of the Update message is processed normally.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 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	**nexthop trigger-delay** { **critical** *delay* | **non-critical** *delay* } **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# nexthop trigger-delay critical 15000`	Sets the critical next-hop trigger delay.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `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** *as-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 206`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 5	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `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 6	**next-hop-self** **Example:**	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

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# next-hop-self`	advertising a route causes all routes to be advertised with the local network device as the next hop.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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 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** | **ipv6** {**labeled-unicast** | **unicast**}}
6. Use one of these commands:
   - **send-community-ebgp**
   - **send-extended-community-ebgp**
7. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `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** *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 5	**address-family**{**ipv4** {**labeled-unicast** | **unicast** | **ipv6** {**labeled-unicast** | **unicast**}} **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv6 unicast`	Enters neighbor address family configuration mode for the specified address family. Use either **ipv4** or **ipv6** address family keyword with one of the specified address family sub mode identifiers. IPv6 address family mode supports these sub modes: • **labeled-unicast** • **unicast** IPv4 address family mode supports these sub modes: • **labeled-unicast** • **unicast** Refer the **address-family (BGP)** command in *BGP Commands* module of *Routing Command Reference for Cisco NCS 6000 Series Routers* for more information on the Address Family Submode support.
Step 6	Use one of these commands: • **send-community-ebgp** • **send-extended-community-ebgp** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# send-community-ebgp` or	Specifies that the router send community attributes or extended community attributes (which are disabled by default for eBGP neighbors) to a specified eBGP neighbor.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# send-extended-community-ebgp`	
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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** | **ipv6** } **unicast** **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4** | **ipv6** } **unicast** **redistribute eigrp** *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4** | **ipv6** } **unicast** **redistribute isis** *process-id* [**level** { **1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4** | **ipv6** } **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** | **ipv6** } **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** | **ipv6** } **unicast redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **address-family** { **ipv4** | **ipv6** } **unicast redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **address-family** { **ipv4** | **ipv6** } **unicast network** { *ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*]
- **neighbor** *ip-address* **remote-as** *as-number* **address-family** { **ipv4** | **ipv6** } **unicast**
- **route-policy** *route-policy-name* { **in** | **out** }

8. Use the **commit** or **end** command.
9. **show bgp** *ip-address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**route-policy** *name* **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** { *cost-extcommunity-set-name* | *cost-inline-extcommunity-set* } [**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** *as-number* **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
	• **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*] • **address-family** { **ipv4** | **ipv6** } **unicast redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4** | **ipv6** } **unicast redistribute eigrp** *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4** | **ipv6** } **unicast redistribute isis** *process-id* [**level** { **1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4** | **ipv6** } **unicast redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]	
Step 7	Do one of the following: • **address-family** { **ipv4** | **ipv6** } **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** | **ipv6** } **unicast redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4** | **ipv6** } **unicast redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4** | **ipv6** } **unicast network** { *ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*] • **neighbor** *ip-address* **remote-as** *as-number* **address-family** { **ipv4** | **ipv6** } **unicast** • **route-policy** *route-policy-name* { **in** | **out** }	
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 9	**show bgp** *ip-address* **Example:** `RP/0/RP0/CPU0:router# show bgp 172.168.40.24`	Displays the cost community in the following format: Cost: *POI* : *cost-community-ID* : *cost-number*

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 99.

Note Storing updates from a neighbor works only if either the neighbor is route refresh capable or the **soft-reconfiguration inbound** command is configured. Even if the neighbor is route refresh capable and the **soft-reconfiguration inbound** command is configured, the original routes are not stored unless the **always** option is used with the command. The original routes can be easily retrieved with a route refresh request. Route refresh sends a request to the peer to resend its routing information. The **soft-reconfiguration inbound** command stores all paths received from the peer in an unmodified form and refers to these stored paths during the clear. Soft reconfiguration is memory intensive.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **address-family** { **ipv4** | **ipv6** } **unicast**
5. **soft-reconfiguration inbound** [**always]**
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **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** *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 4	**address-family** { **ipv4** | **ipv6** } **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 5	**soft-reconfiguration inbound** [**always]** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# soft-reconfiguration inbound always`	Configures the software to store updates received from a specified neighbor. Soft reconfiguration inbound causes the software to store the original unmodified route in addition to a route that is modified or filtered. This allows a "soft clear" to be performed after the inbound policy is changed. Soft reconfiguration enables the software to store the incoming updates before apply policy if route refresh is not supported by the peer (otherwise a copy of the update is not stored). The **always** keyword forces the software to store a copy even when route refresh is supported by the peer.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `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** *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 5	**keychain** *name* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# keychain kych_a`	Configures keychain-based authentication.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions:

	Command or Action	Purpose
		• **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Disabling 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 127`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `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	**shutdown** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# shutdown`	Disables all active sessions for the specified neighbor.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Resetting Neighbors Using BGP Inbound Soft Reset

Perform this task to trigger an inbound soft reset of the specified address families for the specified group or neighbors. The group is specified by the ***** , *ip-address* , *as-number* , or **external** keywords and arguments.

Resetting neighbors is useful if you change the inbound policy for the neighbors or any other configuration that affects the sending or receiving of routing updates. If an inbound soft reset is triggered, BGP sends a REFRESH request to the neighbor if the neighbor has advertised the ROUTE_REFRESH capability. To determine whether the neighbor has advertised the ROUTE_REFRESH capability, use the **show bgp neighbors** command.

SUMMARY STEPS

1. **show bgp neighbors**
2. **clear bgp** { **ipv4 | ipv6**} { **unicast** | **labeled-unicast** } **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	ipv6**} { **unicast**	**labeled-unicast** } **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. • The ***** keyword resets all BGP neighbors. • The *ip-address* argument specifies the address of the neighbor to be reset. • The *as-number* argument specifies that all neighbors that match the autonomous system number be reset. • The **external** keyword specifies that all external neighbors are reset.

Resetting Neighbors Using BGP Outbound Soft Reset

Perform this task to trigger an outbound soft reset of the specified address families for the specified group or neighbors. The group is specified by the ***** , *ip-address* , *as-number* , or **external** keywords and arguments.

Resetting neighbors is useful if you change the outbound policy for the neighbors or any other configuration that affects the sending or receiving of routing updates.

If an outbound soft reset is triggered, BGP resends all routes for the address family to the given neighbors.

To determine whether the neighbor has advertised the ROUTE_REFRESH capability, use the **show bgp neighbors** command.

SUMMARY STEPS

1. **show bgp neighbors**
2. **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	**out** **Example:** `RP/0/RP0/CPU0:router# clear bgp ipv4 unicast 10.0.0.2 soft out`	Soft resets a BGP neighbor. • The ***** keyword resets all BGP neighbors. • The *ip-address* argument specifies the address of the neighbor to be reset. • The *as-number* argument specifies that all neighbors that match the autonomous system number be reset. • The **external** keyword specifies that all external neighbors are reset.

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. }

DETAILED STEPS

	Command or Action	Purpose

Step 1	}
	 Example:
	 `RP/0/RP0/CPU0:router# clear bgp ipv4 unicast 10.0.0.3`
		Clears a BGP neighbor.

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** | **labeled-unicast** } | **ipv6** { **unicast** | **labeled-unicast** }
2. **clear bgp external**
3. **clear bgp ***

DETAILED STEPS

	Command or Action	Purpose
Step 1	**clear bgp** { **ipv4** { **unicast** | **labeled-unicast** } | **ipv6** { **unicast** | **labeled-unicast** } **Example:** `RP/0/RP0/CPU0:router# clear bgp ipv4 172.20.1.1`	Clears a specified neighbor.
Step 2	**clear bgp external** **Example:** `RP/0/RP0/CPU0:router# clear bgp external`	Clears all external peers.
Step 3	**clear bgp *** **Example:** `RP/0/RP0/CPU0:router# clear bgp *`	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	**show bgp cidr-only** **Example:** `RP/0/RP0/CPU0:router# show bgp cidr-only`	Displays routes with nonnatural network masks (classless interdomain routing [CIDR]) routes.
Step 2	**show bgp community** *community-list* [**exact-match**] **Example:** `RP/0/RP0/CPU0:router# show bgp community 1081:5 exact-match`	Displays routes that match the specified BGP community.
Step 3	**show bgp regexp** *regular-expression* **Example:** `RP/0/RP0/CPU0:router# show bgp regexp "^3 "`	Displays routes that match the specified autonomous system path regular expression.
Step 4	**show bgp** **Example:** `RP/0/RP0/CPU0:router# show bgp`	Displays entries in the BGP routing table.
Step 5	**show bgp neighbors** *ip-address* [**advertised-routes** | **dampened-routes** | **flap-statistics** | **performance-statistics** | **received** *prefix-filter* | **routes**] **Example:** `RP/0/RP0/CPU0:router# show bgp neighbors 10.0.101.1`	Displays information about the BGP connection to the specified neighbor. • The **advertised-routes** keyword displays all routes the router advertised to the neighbor. • The **dampened-routes** keyword displays the dampened routes that are learned from the neighbor. • The **flap-statistics** keyword displays flap statistics of the routes learned from the neighbor.

	Command or Action	Purpose
		• The **performance-statistics** keyword displays performance statistics relating to work done by the BGP process for this neighbor. • The **received** *prefix-filter* keyword and argument display the received prefix list filter. • The **routes** keyword displays routes learned from the neighbor.
Step 6	**show bgp paths** **Example:** `RP/0/RP0/CPU0:router# show bgp paths`	Displays all BGP paths in the database.
Step 7	**show bgp neighbor-group** *group-name* **configuration** **Example:** `RP/0/RP0/CPU0:router# show bgp neighbor-group group_1 configuration`	Displays the effective configuration for a specified neighbor group, including any configuration inherited by this neighbor group.
Step 8	**show bgp summary** **Example:** `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 process detail**
4. **show bgp summary**
5. **show placement program bgp**
6. **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.

	Command or Action	Purpose
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 process detail** **Example:** `RP/0/RP0/CPU0:router# show bgp processes detail`	Displays detailed process information including the memory used by each of various internal structure types.
Step 4	**show bgp summary** **Example:** `RP/0/RP0/CPU0:router# show bgp summary`	Displays the status of all BGP connections.
Step 5	**show placement program bgp** **Example:** `RP/0/RP0/CPU0:router# show placement program bgp`	Displays BGP program information. • If a program is shown as having 'rejected locations' (for example, locations where program cannot be placed), the locations in question can be viewed using the **show placement program bgp** command. • If a program has been placed but not started, the amount of elapsed time since the program was placed is displayed in the Waiting to start column.
Step 6	**show placement program brib** **Example:** `RP/0/RP0/CPU0:router# show placement program brib`	Displays bRIB program information. • If a program is shown as having 'rejected locations' (for example, locations where program cannot be placed), the locations in question can be viewed using the **show placement program bgp** command. • If a program has been placed but not started, the amount of elapsed time since the program was placed is displayed in the Waiting to start column.

Monitoring BGP Update Groups

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

SUMMARY STEPS

1. **show bgp update-group** [**neighbor** *ip-address* | *process-id.index* [**summary** | **performance-statistics**]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp update-group** [**neighbor** *ip-address* | *process-id.index* [**summary** | **performance-statistics**]] **Example:** `RP/0/RP0/CPU0:router# show bgp update-group 0.0`	Displays information about BGP update groups. • The *ip-address* argument displays the update groups to which that neighbor belongs. • The *process-id.index* 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Specifies the BGP AS number, and enters the BGP configuration mode, for configuring BGP routing processes.
Step 3	**nsr** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# nsr`	Activates BGP Nonstop routing.

	Command or Action	Purpose
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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** | **ipv6 unicast** }
8. **additional-paths receive**
9. **additional-paths send**
10. **additional-paths selection route-policy** *route-policy-name*
11. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**route-policy** *route-policy-name* **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:**	Decides the actions and dispositions for the given route.

	Command or Action	Purpose

	`RP/0/RP0/CPU0:router (config-rpl)#if community matches-any (*) then`
	`  set path-selection all advertise`
	`  else`	

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** *as-number* **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** | **ipv6 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.
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** *route-policy-name* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**vrf** *vrf_name* **Example:** `RP/0/RSP0/CPU0:PE51_-9010(config)#vrf vrf_1`	Enters VRF configuration mode.
Step 3	**address-family** {**ipv4** | **ipv6**} **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: • **import from default-vrf route-policy** *route-policy-name* [**advertise-as-vpn**]	Imports routes from default-VRF to non-default VRF or from non-default VRF to default-VRF. • **import from default-vrf**—configures import from default-VRF to non-default-VRF.

	Command or Action	Purpose

	• **export to default-vrf route-policy** *route-policy-name*
	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`
		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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

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.

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-fallover disable
  bgp confederation peers
    65001
    65002
  bgp confederation identifier 1
  bgp router-id 192.0.2.1
  address-family ipv4 unicast
    aggregate-address 10.2.0.0/24 route-policy set_next_hop_agg_v4
    aggregate-address 10.3.0.0/24
    redistribute static route-policy set_next_hop_static_v4

  address-family ipv6 unicast
    aggregate-address 2012::/64 route-policy set_next_hop_agg_v6
    aggregate-address 2013::/64
    redistribute static route-policy set_next_hop_static_v6

  neighbor 10.0.101.60
    remote-as 65000
    address-family ipv4 unicast

  neighbor 10.0.101.61
    remote-as 65000
    address-family ipv4 unicast

  neighbor 10.0.101.62
    remote-as 3
    address-family ipv4 unicast
      route-policy pass-all in
      route-policy pass-all out

  neighbor 10.0.101.64
    remote-as 5
    update-source Loopback0
    address-family ipv4 unicast
      route-policy pass-all in
```

```
        route-policy pass-all out
```

Displaying BGP Update Groups: Example

The following is sample output from the **show bgp update-group** command run in XR EXEC 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..0.0 255..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. .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:

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

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
```

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
```

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.

Flow tag can be set for prefixes of routing attributes if the flow tag is set to a value other than zero. By default, the value of a flow tag is set to zero for all prefixes unless you modify it. Ucode does not launch TCAM lookup to improve packet forwarding performance for the prefixes for which flow tag value is zero. Therefore, traffic for the prefixes which has flow tag set to zero does not hit class default and can be used for subnetting.

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.
- Only destination-based flow-tag propagation is supported.

Configuring Destination-Based Flow-Tag Propagation

The destination-based flow tag feature allows you to match packets based on the flow-tag assigned to the destination address of the incoming packets. Once matched, you can then apply any supported PBR action on this policy.

Note You will not be able to enable both QPPB and flow tag features simultaneously on an interface.

Configuration

Use the following sample configuration to configure destination-based flow-tag propagation.

```
/* Configure a route policy for flow-tag propagation */
Router(config)# prefix-set FLOWTAG36
Router(config-pfx)# 10.1.30.0/24
Router(config-pfx)# end-set
Router(config)# prefix-set FLOWTAG38
Router(config-pfx)# 10.1.40.0/24
Router(config-pfx)# end-set

Router(config)# route-policy SETFLOWTAG
Router(config-rpl)# if destination in FLOWTAG36 then set flow-tag 36 endif
Router(config-rpl)# if destination in FLOWTAG38 then set flow-tag 38 endif
Router(config-rpl)# end-policy
Router(config)# commit
Tue Apr  3 15:10:07.223 IST

/* Configure the class map and policy map for flow-tag propagation */
Router(config)# class-map type traffic match-any FLOWMATCH36
Router(config-cmap)# match flow-tag 36
Router(config-cmap)# end-class-map

Router(config)# class-map type traffic match-any FLOWMATCH38
Router(config-cmap)# match flow-tag 38
Router(config-cmap)# end-class-map

Router(config)# policy-map type pbr FLOWMATCH
Router(config-pmap)# class type traffic FLOWMATCH36
Router(config-pmap-c)# redirect ipv4 nexthop 20.20.20.1
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic FLOWMATCH38
Router(config-pmap-c)# drop
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic DEFAULT
Router(config-pmap-c)# exit
Router(config-pmap)# end-policy-map

/* Configure BGP with flow-tag propagation */
 Router(config)# router bgp 10
 Router(config-bgp)# bgp router-id 1.1.1.1
 Router(config-bgp)# address-family ipv4 unicast
 Router(config-bgp-af)# table-policy SETFLOWTAG
 Router(config-bgp-af)# redistribute static
 Router(config-bgp-af)# bgp attribute-download
 Router(config-bgp-af)# redistribute connected
 Router(config-bgp-af)# exit
```

```
 Router(config-bgp)# neighbor 20.20.20.1/24
 Router(config-bgp-nbr)# remote-as 20
 Router(config-bgp-nbr)# address-family ipv4 unicast
 Router(config-bgp-nbr-af)# route-policy BGPIN in
 Router(config-bgp-nbr-af)# route-policy BGPOUT out
 Router(config-bgp-nbr-af)# exit
 Router(config-bgp-nbr)# exit
 Router(config-bgp)# exit

 Router(config)# route-policy BGPIN
 Router(config-rpl)# pass
 Router(config-rpl)# end-policy
 Router(config)# route-policy BGPOUT
 Router(config-rpl)# pass
 Router(config-rpl)# end-policy

/* Enter the interface configuration mode and enable flow tag on an interface. */
Router(config)# interface GigabitEthernet 0/0/0/1
Router(config-if)# ipv4 address 10.10.10.1 255.255.255.0
Router(config-if)# service-policy type pbr input FLOWMATCH
Router(config-if)# no shut

/* Commit the configuration */
Router(config-if)# commit
Mon Mar 19 07:59:01.081 IST
RP/0/0/CPU0:Mar 19 07:59:01.537 : ifmgr[403]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/1, changed state to Down
RP/0/0/CPU0:Mar 19 07:59:01.619 : ifmgr[403]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/1, changed state to Up

 /* Validate the configuraton */
Router(config)# do show run
Mon Mar 19 08:03:31.106 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Mon Mar 19 08:02:55 2018 by UNKNOWN
…
class-map type traffic match-any FLOWMATCH36
 match flow-tag 36
 end-class-map
!
!
class-map type traffic match-any FLOWMATCH40
 match flow-tag 40
 end-class-map
!
policy-map type pbr FLOWMATCH
 class type traffic FLOWMATCH36
  transmit
 !
 class type traffic FLOWMATCH40
  transmit
 !
 class type traffic class-default
 !
 end-policy-map
!
interface GigabitEthernet0/0/0/0
 ipv4 forwarding-enable
 ipv6 address 2000::2/64
!
interface GigabitEthernet0/0/0/1
 service-policy type pbr input FLOWMATCH
 ipv4 address 10.10.10.1 255.255.255.0
```

```
!
interface GigabitEthernet0/0/0/2
 ipv4 forwarding-enable
 ipv6 address 3000::2/64
!
…
!
prefix-set FLOWTAG36
  10.1.30.0/24
end-set
!
prefix-set FLOWTAG40
  10.1.40.0/24
end-set
!
route-policy SETFLOWTAG
  if destination in FLOWTAG36 then
    set flow-tag 36
  endif
  if destination in FLOWTAG40 then
    set flow-tag 40
  endif
end-policy
!
!
router bgp 10
bgp router-id 1.1.1.1
address-family ipv4 unicast
table-policy SETFLOWTAG
redistribute static
bgp attribute-download
redistribute connected
!
neighbor 20.20.20.1/24
remote-as 20
address-family ipv4 unicast
route-policy BGPIN in
route-policy BGPOUT out
!
route-policy BGPIN
 pass
end-policy
route-policy BGPOUT
 pass
end-policy
!
```

You have successfully configured destination-based flow-tag propagation.

Where to Go Next

For detailed information about BGP commands, see *Routing Command Reference for Cisco NCS 6000 Series 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	*Routing Command Reference for Cisco NCS 6000 Series Routers*
Cisco Express Forwarding (CEF) commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*IP Addresses and Services Command Reference for Cisco NCS 6000 Series Routers*
Bidirectional Forwarding Detection (BFD)	*Interface and Hardware Component Configuration Guide for Cisco NCS 6000 Series Routers* and *Interface and Hardware Component Command Reference for the Cisco NCS 6000 Series Routers*
Task ID information.	Configuring AAA Services on module of *System Security Configuration Guide for Cisco NCS 6000 Series Routers*

Standards

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

MIBs

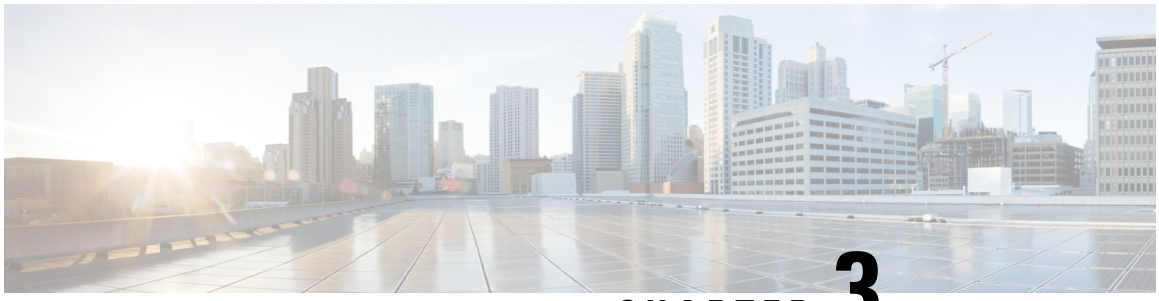
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

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
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 4724	*Graceful Restart Mechanism for BGP*

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 NCS 6000 Series Routers*.

Feature History for Implementing BGP Flowspec

Release 5.2.4	This feature was introduced.

• BGP Flow Specification, on page 121

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 123 provides details on flow specifications, supported matching criteria and traffic filtering action.

The BGP Flowspec feature cannot coexist with MAP-E and PBR on a given interface. If you configure BGP Flowspec with PBR, the router does not display any error or system message. The router ignores the BGP-FS configuration and the feature will not function.

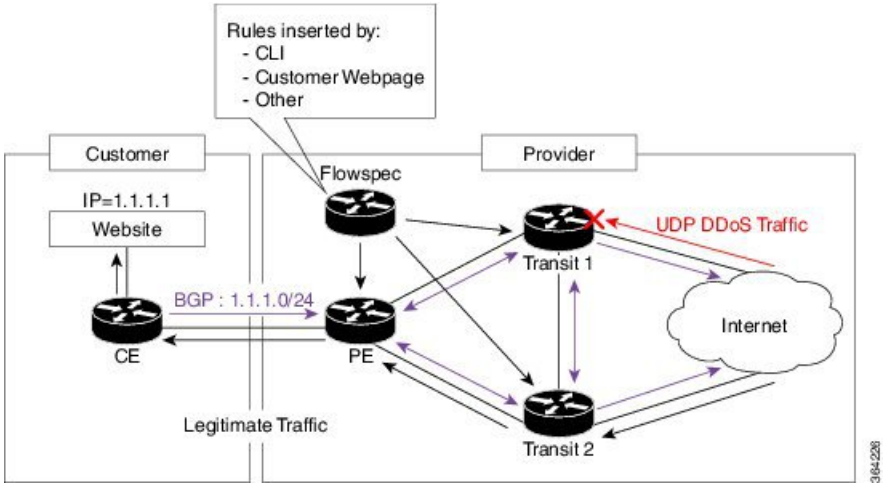
Limitations

These limitations apply for BGP flowspec:

- For in-service software upgrade (ISSU), there is no support for zero packet loss (ZPL) for BGP Flowspec traffic.
- 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.
- QoS takes precedence over BGP flowspec.

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 cm1
     match source-address ipv4 100.0.0.0/24

policy-map type pbr pm1
    class type traffic cm1
      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 How to Configure BGP Flowspec, on page 130.

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 124 and Traffic Filtering Actions, on page 127 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:

Table 5: Tuple definition possibilities

BGP Flowspec NLRI type	QoS match fields	Description and Syntax Construction	Value input method
Type 1	IPv4 Destination address	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. Encoding: <type (1 octet), prefix length (1 octet), prefix> **Syntax:** **match destination-address** {**ipv4**} *address/mask length*	Prefix length
Type 2	IPv4 Source address	Defines the source prefix to match. Encoding: <type (1 octet), prefix-length (1 octet), prefix> **Syntax:** **match source-address** {**ipv4**} *address/mask length*	Prefix length

Type 3	IPv4 last next header Protocol	Contains a set of {operator, value} pairs that are used to match the IP protocol value byte in IP packets. Encoding: <type (1 octet), [op, value]+> **Syntax:** Type 3: **match protocol** {*protocol-value* | *min-value* *−max-value*}	Multi value range
Type 4	IPv4 source or destination port	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 in unable to locate the transport header. Encoding: <type (1 octet), [op, value]+> **Syntax:** **match source-port** {*source-port-value* | *min-value* *−max-value*} **match destination-port** {*destination-port-value* | *min-value* *−max-value*}	Multi value range
Type 5	IPv4 destination port	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. Encoding: <type (1 octet), [op, value]+> **Syntax:** **match destination-port** {*destination-port-value* |[*min-value* *− max-value*]}	Multi value range
Type 6	IPv4 Source port	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. Encoding: <type (1 octet), [op, value]+> **Syntax:** **match source-port** {*source-port-value* |[*min-value* *− max-value*]}	Multi value range

Type 7	IPv4 ICMP type	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. Encoding: <type (1 octet), [op, value]+> **Syntax:** **match** {**ipv4**} **icmp-type** {*value* | *min-value* **-** *max-value*}	Single value **Note** Multi value range is not supported.
Type 8	IPv4 ICMP code	Defines a list of {operation, value} pairs used to match the code field of an ICMP packet. Values are encoded using a single byte. Encoding: <type (1 octet), [op, value]+> **Syntax:** **match** {**ipv4**} **icmp-code** {*value* | *min-value* **-** *max-value*}	Single value **Note** Multi value range is not supported.
Type 9	IPv4 TCP flags (2 bytes include reserved bits) **Note** Reserved and NS bit not supported	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. Encoding: <type (1 octet), [op, bitmask]+> **Syntax:** **match tcp-flag** *value* **bit-mask** *mask_value*	Bit mask
Type 10	IPv4 Packet length	Match on the total IP packet length (excluding Layer 2, but including IP header). Values are encoded using 1- or 2-byte quantities. Encoding: <type (1 octet), [op, value]+> **Syntax:** **matchpacket length** {*packet-length-value* | *min-value* **-** *max-value*}	Multi value range

Type 11	IPv4 DSCP	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. Encoding: <type (1 octet), [op, value]+> **Syntax:** **match dscp** {*dscp-value* | *min-value* **-** *max-value*}	Multi value range
Type 12	IPv4 Fragmentation bits **Note** IPv6 BGP flowspec does not supports Type 12 NRLI.	Identifies a fragment-type as the match criterion for a class map. Encoding: <type (1 octet), [op, bitmask]+> **Syntax:** **match fragment type** [**is-fragment**]	Bit mask

In a given flowspec rule, multiple action combinations can be specified without restrictions.

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

Traffic Filtering Actions, on page 127 provides information on the actions that can be associated with a flow. How to Configure BGP Flowspec, on page 130 explains the procedure to configure BGP flowpsec 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. You can use the following extended community values to specify particular actions:

Type	Extended Community	PBR Action	Description

0x8006	traffic-rate 0 traffic-rate <rate>	Drop Police	Traffic-rate extended community is a non-transitive extended community across the autonomous system boundary. It uses the following extended community encoding: You can assign the first two octets that carry the 2-octet id from a 2-byte autonomous system number. When a 4-byte autonomous system number is locally present, you can use the 2 least significant bytes of such an autonomous system number. 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 causes discarding of all traffic for the particular flow. **Command syntax** police rate < > | drop
0x8008	redirect-vrf	Redirect VRF	Redirect extended community redirects the traffic 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 local matter (for example, you can elect the instance with the lowest Route Distinguisher value). This extended community uses the same encoding as the Route Target extended community [RFC4360]. **Command syntax based on route-target** redirect {ipv4} extcommunity rt <route_target_string>
0x8009	traffic-marking	Set DSCP	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. RFC 5575 encodes the extended community as a sequence. It is a sequence of 5 zero bytes followed by the DSCP value encoded in the 6 least significant bits of the sixth byte. **Command syntax** set dscp <6-bit value> set ipv4 traffic-class <8-bit value>

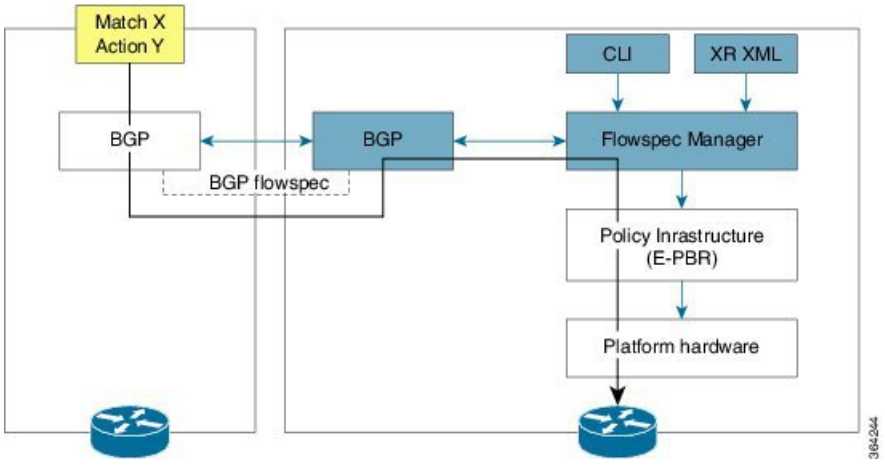
0x0800	Redirect IP NH	Redirect IPv4 Nexthop	Redirect IP Next-Hop extended community announces the availability of one or more flowspec Network Layer Reachability Information (NLRI). When a BGP speaker receives an UPDATE message with the redirect-to-IP extended community, it creates a traffic-filtering rule for every flow-spec NLRI in the message that has this path as its best path. The filter entry matches the IP packets that BGP describes in the NLRI field. BGP specifies an IPv4 address in the Network Address of Next-Hop field of the associated Multiprotocol Reachable NLRI (MP_REACH_NLRI) The filter entry redirects the IP packets or copies them toward that address. **Note** The redirect-to-IP extended community is valid with any other set of flow-spec extended communities. If the set includes a redirect-to-VRF extended community (type 0x8008), the filter ignores the redirect-to-IP extended community. **Command syntax** redirect {ipv4} next-hop <ipv4 address> {ipv4 address}

Configure a Class Map, on page 132 explains how you can configure specific match criteria for a class map.

BGP Flowspec Client-Server (Controller) Model and Configuration

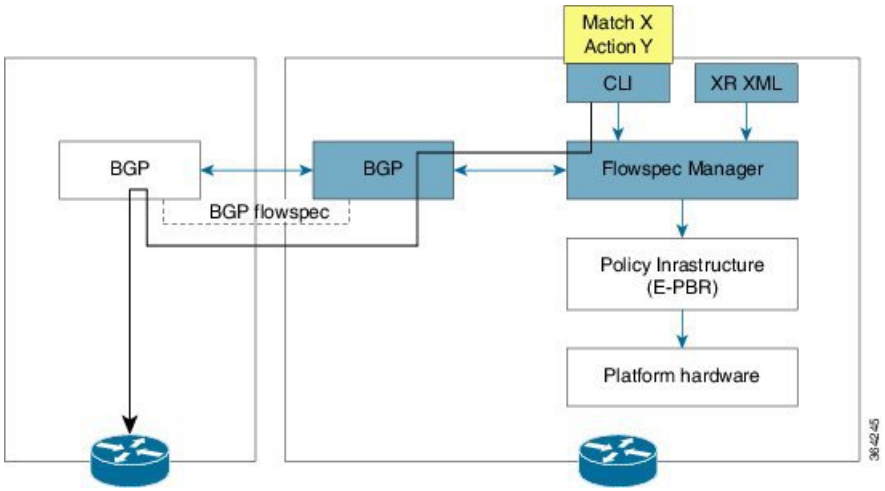
The BGP Flowspec model comprises of a Client and a Server (Controller). The Controller is responsible for sending or injecting the flowspec NRLI entry. The client (acting as a BGP speaker) receives that NRLI and programs the hardware forwarding to act on the instruction from the Controller. An illustration 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 (Enhance Policy-based Routing) infrastructure, which in turn programs the hardware from the underlaying 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 Flowspec on BGP Side, on page 131 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:

  - Define Policy Map, on page 133
  - Configure a Class Map, on page 132
  - Link Flowspec to PBR Policies , on page 135

How to Configure BGP Flowspec

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

- Enable Flowspec on BGP Side, on page 131
- Configure a Class Map, on page 132
- Define Policy Map, on page 133
- Link Flowspec to PBR Policies , on page 135

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

Enable Flowspec on BGP Side

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** | **vpnv4** } **flowspec**
4. **exit**
5. **neighbor** *ip-address*
6. **remote-as** *as-number*
7. **address-family** { **ipv4** } **flowspec**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 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** | **vpnv4** } **flowspec** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 flowspec`	Specifies either the IPv4, vpn4 address family and enters address family configuration submode, and initializes the global address family for flowspec policy mapping.
Step 4	**exit** **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# exit`	Returns the router to BGP configuration mode.
Step 5	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)#neighbor 192.0.2.1`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 6	**remote-as** *as-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)#remote-as 100`	Assigns a remote autonomous system number to the neighbor.

	Command or Action	Purpose
Step 7	**address-family** { **ipv4** } **flowspec** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 flowspec`	Specifies an address family and enters address family configuration submode, and initializes the global address family for flowspec policy mapping.

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]** *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map type traffic match all classc1`	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** *match-statement* **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match protocol ipv4 1 60`	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: • Type 1: **match destination-address** {**ipv4**} *address/mask length* • Type 2: **match source-address** {**ipv4**} *address/mask length* • Type 3: **match protocol** {*protocol-value* | *min-value* *−max-value*} • Type 4: Create two class-maps: one with source-port and another with destination-port: • **match source-port** {*source-port-value* | *min-value* *−max-value*}

	Command or Action	Purpose
		• **match destination-port** {*destination-port-value* | *min-value* −*max-value*} **Note** These are applicable only for TCP and UDP protocols. • Type 5: **match destination-port** {*destination-port-value* |[*min-value* − *max-value*]} • Type 6: **match source-port** {*source-port-value* |[*min-value* − *max-value*]} • Type 7: **match** {**ipv4**}**icmp-code** {*value* | *min-value* −*max-value*} • Type 8: **match**{**ipv4**}**icmp-type** {*value* | *min-value* −*max-value*} • Type 9: **match tcp-flag** *value* **bit-mask** *mask_value* • Type 10: **matchpacket length** {*packet-length-value* | *min-value* −*max-value*}} • Type 11: **match dscp** {*dscp-value* | *min-value* −*max-value*} • Type 12: **match fragment-type** { **is-fragment** } • Type 13: **match ipv4 flow-label** {*value* | *min-value* −*max-value*} >
Step 4	**end-class-map** **Example:** `RP/0/RP0/CPU0:router(config-cmap)# end-class-map`	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 .

Define 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 132 .

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** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**policy-map type pbr** *policy-map* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map type pbr policyp1`	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	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.
Step 4	**class type traffic** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class type traffic classc1`	Associates a previously configured traffic class with the policy map, and enters control policy-map traffic class configuration mode.
Step 5	*action* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set dscp 5`	Define extended community actions as per your requirement. The options include: • Traffic rate: **police rate** *rate* • Redirect VRF: **redirect** { **ipv4** } **extcommunity rt** *route_target_string* • Traffic Marking: **set** { **dscp** *rate* | **destination-address** {**ipv4**} *8-bit value*} • Redirect IP NH: **redirect** { **ipv4** } **nexthop** *ipv4 address* { *ipv4 address*}

	Command or Action	Purpose
		Traffic Filtering Actions, on page 127 provides conceptual information on these extended community actions.
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 7	**end-policy-map** **Example:** `RP/0/RP0/CPU0:router(config-cmap)# end-policy-map`	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 Flowspec to PBR Policies , on page 135

Link Flowspec to PBR Policies

For BGP flowspec, a PBR 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 PBR policy on an interface, it will not be overwritten by the BGP flowspec policy. If you remove the policy from an interface, PBR infrastructure will automatically apply BGP flowspec policy on it, if one was active at the VRF level.

Note At a time only one PBR 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. Use the **commit** or **end** command.
8. **exit**
9. **show flowspec** { **afi-all** | **client** | **ipv4** | **summary** | **vrf**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**flowspec** **Example:** `RP/0/RP0/CPU0:router(config)# flowspec`	Enters the flowspec configuration mode.
Step 3	**local-install interface-all** **Example:** `RP/0/RP0/CPU0:router(config-flowspec)# local-install interface-all`	(Optional) Installs the flowspec policy on all interfaces.
Step 4	**address-family ipv4** **Example:** `RP/0/RP0/CPU0:router(config-flowspec)# address-family ipv4`	Specifies either an IPv4 address family and enters address family configuration submode.
Step 5	**local-install interface-all** **Example:** `RP/0/RP0/CPU0:router(config-flowspec-af)# local-install interface-all`	(Optional) Installs the flowspec policy on all interfaces under the subaddress family.
Step 6	**service-policy type pbr** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config-flowspec-af)# service-policy type pbr policys1`	Attaches a policy map to an IPv4 interface to be used as the service policy for that interface.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-flowspec-vrf-af)# exit`	Returns the router to flowspec configuration mode.
Step 9	**show flowspec** { **afi-all** | **client** | **ipv4** | **summary** | **vrf** **Example:** `RP/0/RP0/CPU0:routershow 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** { **afli-all** | **ipv4** }
4. **show bgp ipv4 flowspec**
5. **show pbr-pal ipolicy all location***node-id*

DETAILED STEPS

	Command or Action	Purpose

Step 1	**show processes flowspec_mgr location all**
	Example:
	`# 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`
		Specifies whether the flowspec process is running on your system or not. The flowspec manager is responsible for creating, distributing and installing the flowspec rules on the hardware.

	Command or Action	Purpose
	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	
Step 2	**show flowspec summary** **Example:** # show flowspec summary FlowSpec Manager Summary: Tables: 2 Flows: 1 RP/0/3/CPU0:RA01_R4#	Provides a summary of the flowspec rules present on the entire node. In this example, IPv4 has been enabled, and a single flow has been defined across the entire table.
Step 3	**show flowspec vrf** *vrf_name* | **all** { **afli-all** | **ipv4** } **Example:** # 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 all afi-all summary Flowspec VRF+AFI table summary: VRF: default AFI: IPv4 Total Flows: 1 Total Service Policies: 1 --- # 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	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. 'VRF all' displays information across all the VRFs configured on the table and afli-all displays information for all address families. 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.

	Command or Action	Purpose
	Transmitted : 0/0 Dropped : 0/0	

Step 4	**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`
		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.

Step 5	**show pbr-pal ipolicy all location***node-id*	On platform dependent devices, use this command to verify if a flowspec rule configured on the controller router is available on the BGP side.

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** }
5. **flowspec next-hop unchanged**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `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** } **Example:** `RP/0/RP0/CPU0:router(config-bgp)# router bgp 100 neighbor 1.1.1.1 address-family ipv4`	Specifies the IPv4 address family and enters address family configuration submode, and initializes the global address family.
Step 5	**flowspec next-hop unchanged** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# router bgp 100 neighbor 1.1.1.1 address-family ipv4 flowspec next-hop unchanged`	Preserves the next-hop for the flowspec unchanged.

Validate BGP Flowspec

BGP Flowspec validation is enabled by default for flowspec SAFI routes for IPv4. 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 142 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** } **flowspec disable**
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters XR Config mode.

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** } **flowspec disable**

Example:

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

Disable flowspec policy on the selected interface.

Step 4 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

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] 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** }
5. **flowspec validation** { **disable** | **redirect disable** }

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

	Command or Action	Purpose
Step 3	**neighbor** *ip-address* **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** } **Example:** `RP/0/RP0/CPU0:router(config-bgp)# router bgp 100 neighbor 1.1.1.1 address-family ipv4`	Specifies the IPv4 address family and enters address family configuration submode, and initializes the global address family.
Step 5	**flowspec validation** { **disable** | **redirect disable** } **Example:** `RP/0/RP0/CPU0:router(config-bgp)# router bgp 100 neighbor 1.1.1.1 address-family ipv4 flowspec validation disable`	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.

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	*Routing Command Reference for Cisco NCS 6000 Series Routers*

Standards

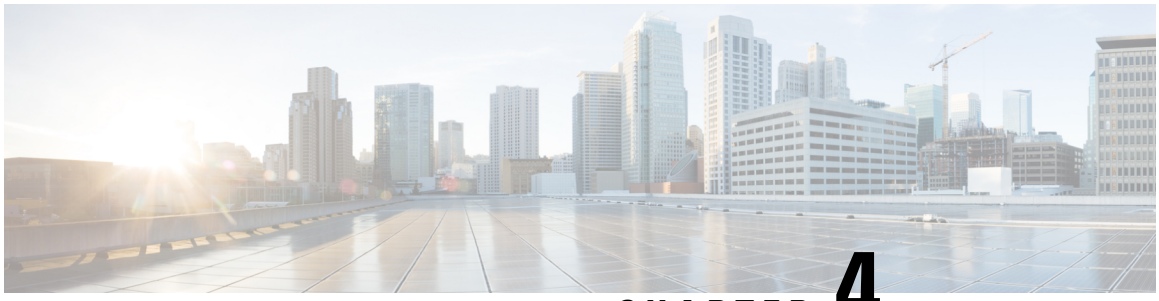
Standards	Title
draft-ietf-idr-flowspec-redirect-ip-01	BGP Flow-Spec Redirect to IP Action
draft-simpson-idr-flowspec-redirect-02	BGP Flow-Spec Extended Community for Traffic Redirect to IP Next Hop
draft-ietf-idr-bgp-flowspec-oid-02	Revised Validation Procedure for BGP Flow Specifications

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 NCS 6000 Series 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 4.3.1	Support for these features was added: • BFD over MPLS Traffic Engineering LSPs

Release	Modification
Release 5.0.0	This feature was introduced.
Release 5.2.5	Support for BFD over Logical Bundle was added.

- Prerequisites for Implementing BFD, on page 147
- Restrictions for Implementing BFD, on page 148
- Information About BFD, on page 149
- How to Configure BFD, on page 164
- Configuration Examples for Configuring BFD, on page 196
- Where to Go Next, on page 205
- Additional References, on page 206

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.
- 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)
  - 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:
-
- 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.
- SNMP traps are not supported for multipath BFD sessions.

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:

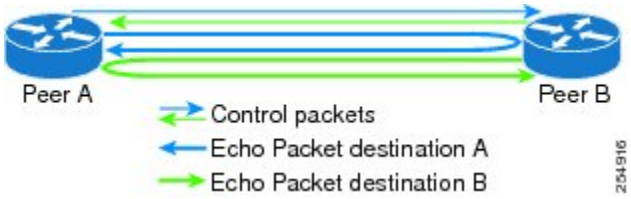
- Each system periodically sends BFD control packets to one another. Packets sent by BFD router "Peer A" to BFD router "Peer B" have a source address from Peer A and a destination address for Peer B.
- Control packet streams are independent of each other and do not work in a request/response model.
- If a number of packets in a row are not received by the other system, the session is declared down.

Figure 8: 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 9: 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* x *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* x *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* x *M*) or (*I* x *M* x *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* x *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. 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* x *M* x *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 roundtip 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* x *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* x *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* x *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.

```
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 x 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 6: 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* x *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* x *M*) x *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 6: 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 Detection Tim (Echo Interva 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

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).

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 QoSConfiguration Guide for Cisco NCS 6000 Series 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)
- 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)
  - Logical interfaces such as bundles, GRE, PWHE

Note BFD is supported on the above interface types and not on logical interfaces unless specifically stated. For example, BFD cannot be configured on BVI and interflex.

- 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

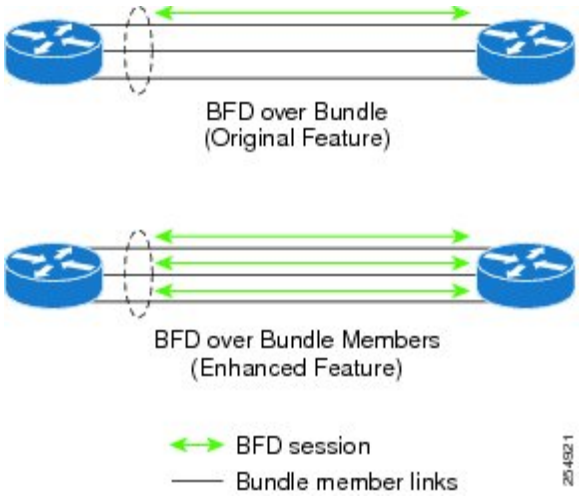
BFD on Bundled VLANs

Note For more information on configuring a VLAN bundle, see the module.

BFD Over Member Links on Link Bundles

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 10: 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 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 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.

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, MPLS Traffic Engineering, and LDP. The Cisco IOS XR Software BFD MHv6 implementation is in accordance with *IETF RFC5883 for IPv6 networks*.

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 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 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.

BFD over Bundle and BFD over Logical Bundle

Link Aggregation Control Protocol (LACP) allows a network device to negotiate an automatic bundling of links by sending LACP packets to their directly connected peer. LACP provides a keep-alive mechanism for the link members. While the default keep-alive is 30s, it is configurable to up to 1s. LACP can detect failures on a per-physical-member link. However, the LACP timers do not fulfill the criteria of current fast convergence requirements.

Differences between BFD over Bundle and BFD over Logical Bundle

BFD over Bundle (BoB) (RFC 7130) has a BFD session on each bundle member. The client is the bundle manager. If a BFD session goes down on a specific member link, the whole bundle interface goes down. That is, when the member link goes down, the number of available links falls below the required minimum. Hence the routing session is brought down.

BFD over Logical Bundle (BLB) (RFC 5880) treats a bundle interface with all its members as a single interface. BLB is a multipath (MP) single-hop session. If BLB is configured on a bundle there is only one single BFD session that is active. This implies that only one bundle member is being monitored by BFD at any given time.

The client is one of the routing protocols. When BFD detects a failure, the client brings down the routing session.

The mode (BoB or BLB) is determined by how you configure BFD:

- You can enable BoB by configuring BFD under a Bundle-Ether interface.
- You can enable BLB by configuring BFD under a Bundle-Ether interface on a routing client.

Link Aggregation Control Protocol (LACP) allows a network device to negotiate an automatic bundling of links by sending LACP packets to their directly connected peer. LACP provides a keep-alive mechanism for the link members. While the default keep-alive is 30s, it is configurable to up to 1s. LACP can detect failures on a per-physical-member link. However, the LACP timers do not fulfill the criteria of current fast convergence requirements.

BFD over Bundle

BFD over Bundle

BFD Over Bundle (BoB) (RFC 7130) has a BFD session on each bundle member. BOB verifies the ability for each member link to be able to forward Layer 3 packets.

For BFD over Bundle, the BFD client is bundlemgr. When BFD detects a failure on a bundle member, bundlemgr removes that member from the bundle. If there are not enough members to keep the bundle up, then the main Bundle-Ether interface will go down so that all routing protocols running on the main bundle interface or a subinterface will detect an interface down.

BoB does not provide a true Layer 3 check and is not supported on subinterfaces. However, subinterfaces will go down at the same time as the main interface.

BoB is a standard-based fast failure detection of link aggregation (LAG) member links that is interoperable between different platforms. Cisco ASR 9000 support both IETF mode and Cisco mode.

Configure BFD Over Bundle

Perform the following tasks to configure the BOB feature:

- Enable BFD sessions on bundle members
- Specify the BFD destination address on a bundle
- Configure the minimum thresholds for maintaining an active bundle
- Configure BFD packet transmission intervals and failure detection times on a bundle

  Configure BFD over bundles IETF mode support on a per-bundle basis

Note In software mode, it is recommended to use greater than or equal to 150ms as the minimum timer interval.

```
/* Enable BFD sessions on bundle members */
Router(config)# interface Bundle-Ether 1
Router(config-if)# bfd address-family ipv4 fast-detect
Router(config-if)# bfd mode ietf
```

```
/* Specify the BFD destination address on a bundle */
Router(config)# interface Bundle-Ether 1
Router(config-if)# bfd address-family ipv4 destination 10.20.20.1

/* Configure the minimum thresholds for maintaining an active bundle */
Router(config)# interface Bundle-Ether 1
Router(config-if)# bundle minimum-active bandwidth 580000
Router(config-if)# bundle minimum-active links 2

/* Configure BFD packet transmission intervals and failure detection times on a bundle */
Router(config)# interface Bundle-Ether 1
Router(config-if)# bfd address-family ipv4 minimum-interval 2000
Router(config-if)# bfd address-family ipv4 multiplier 30

/* Configure BFD over bundles IETF mode support on a per-bundle basis.  */
/* Alternatively, you can configure Cisco mode. */
Router(config)# interface Bundle-Ether 1
Router(config-if)# bfd mode ietf
Router(config-if)# bfd address-family ipv4 fast-detect
```

Bidirectional Forwarding Detection over Logical Bundle

BFD over Logical Bundle

The BLB feature implements and deploys BFD over bundle interfaces based on RFC 5880. In the BLB, the bundle interface is a single interface, whereas, in BOB, BFD is implemented per member link. BLB is a multipath (MP) single-hop session so at least one line card must be configured under the **bfd multipath** command before a BLB session can come up. Because BFD treats the bundle as a single big interface, BLB requires limited knowledge of the bundle interfaces on which the sessions run. BLB requires information about IP addresses, interface types, and caps on bundle interfaces only. Information such as a list of bundle members, member states, and configured minimum or maximum bundle links are not required. In the case of BLB, the BFD client is not the bundle link but protocols running over the bundle link. In BLB, the BFD client is not bundlemgr but the protocols running over bundle link. 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.

Configuration Example

- Create VLAN subinterface under bundle interface
- Enable BFD on a static route
- Enable BFD on IS-IS
- Enable BFD for OSPF on an interface
- Enable BFD on a BGP neighbor
- Configure multipath capability under BFD

```
/* Create VLAN subinterface under bundle interface */
Router# configure
Router(config)# interface Bundle-Ether 2.1
```

```
Router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
Router(config-if)# encapsulation dot1q 1
Router(config-if)# end

/* Enable BFD on a static route. */
Router# configure
Router(config)# router static
Router(config-static)# address-family ipv4 unicast
Router(config-static)# 10.158.3.13/32 10.1.1.2 bfd fast-detect minimum-interval 300 multiplier
 3

/* Enable BFD on IS-IS. */
Router# configure
Router(config)# router isis cybi
Router(config-isis)# interface Bundle-Ether 2.1
Router(config-isis-if)# bfd minimum-interval 300
Router(config-isis-if)# bfd multiplier 3
Router(config-isis-if)# bfd fast-detect ipv4
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# end

/* Enable BFD for OSPF on an interface. */
Router# configure
Router(config)# router ospf cybi
Router(config-ospf)# area 0
Router(config-ospf)# interface Bundle-Ether 2.1
Router(config-ospf-if)# bfd fast-detect
Router(config-ospf-if)# bfd minimum-interval 300
Router(config-ospf-if)# bfd multiplier 3
Router(config-ospf-if)# end

/* Enable BFD on a BGP neighbor.*/
Router# configure
Router(config)# router bgp 4787
Router(config-bgp)# neighbor 10.158.1.1
Router(config-bgp-nbr)# remote-as 4787
Router(config-bgp-nbr)# update-source Bundle-Ether 2.1
Router(config-bgp-nbr)# bfd fast-detect
Router(config-bgp-nbr)# bfd minimum-interval 300
Router(config-bgp-nbr)# bfd multiplier 3
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy PASS-ALL in
Router(config-bgp-nbr-af)# route-policy PASS-ALL out
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# commit

/* Configure a specific LC (or LCs) to host BLB sessions. The BLB sessions and bundle member
 links need not be configured on the same LC. For example, you can configure the bundle
member links on LC slot 2 and slot 3 while you configure BLB sessions to be hosted on LC
slot 5. */
Router(config)# bfd
Router(config-bfd)# multipath include location 0/6/CPU0
Router(config-bfd)# multipath include location 0/2/CPU0
```

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.
- 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *autonomous-system-number* **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 XR EXEC mode to obtain the *autonomous-system-number* for the current router.
Step 3	**bfd minimum-interval** *milliseconds* **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** *multiplier* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bfd multiplier 7`	Sets the BFD multiplier.
Step 5	**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. This example configures the IP address 172.168.40.24 as a BGP peer.
Step 6	**remote-as** *autonomous-system-number* **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. 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

	Command or Action	Purpose
		between the local networking devices and the neighbor 172.168.40.24.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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, OSFPv3, 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 location***node-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. Use the **commit** or **end** command.
10. **show run router ospf**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

	Command or Action	Purpose
Step 2	**bfd multipath include location***node-id* **Example:** `RP/0/RP0/CPU0:router(config)# bfd multipath include location 0/0/CPU0`	(Optional) Enables BFD multipath for the specified bundle on the interface. This step is required for bundle interfaces. **Note** • This step must be repeated for every line card that has a member link in the bundle interface.
Step 3	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 0`	Enters OSPF configuration mode, allowing you to configure the OSPF routing process. Use the **show ospf** command in XR EXEC mode to obtain the process-name for the current router. **Note** • 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** *milliseconds* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# bfd minimum-interval 6500`	Sets the BFD minimum interval. Range is 15-30000 milliseconds. This example sets the BFD minimum interval to 6500 milliseconds.
Step 5	**bfd multiplier** *multiplier* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# bfd multiplier 7`	Sets the BFD multiplier. This example sets the BFD multiplier to 7.
Step 6	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Configures an Open Shortest Path First (OSPF) area. Replace *area-id* with the OSPF area identifier.
Step 7	**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 *rack/slot/module/port*. • The example indicates a Gigabit Ethernet interface in modular services card slot 3.
Step 8	**bfd fast-detect** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# bfd fast-detect`	Enables BFD to detect failures in the path between adjacent forwarding engines.

	Command or Action	Purpose
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 10	**show run router ospf** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# show run router ospf`	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. Use the **commit** or **end** command.
9. **show run router ospfv3**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters XR Config mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# configure`	
Step 2	**router ospfv3** *process-name* **Example:** `RP/0/RP0/CPU0:routerconfig)# router ospfv3 0`	Enters OSPFv3 configuration mode, allowing you to configure the OSPFv3 routing process. Use the **show ospfv3** command in XR EXEC mode to obtain the process name for the current router. **Note** • 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** *milliseconds* **Example:** `RP/0/RP0/CPU0:router(config-ospfv3)# bfd minimum-interval 6500`	Sets the BFD minimum interval. Range is 15-30000 milliseconds. This example sets the BFD minimum interval to 6500 milliseconds.
Step 4	**bfd multiplier** *multiplier* **Example:** `RP/0/RP0/CPU0:router(config-ospfv3)# bfd multiplier 7`	Sets the BFD multiplier. This example sets the BFD multiplier to 7.
Step 5	**area** *area-id* **Example:** `RP/0//CPU0:router(config-ospfv3)# area 0`	Configures an OSPFv3 area. Replace *area-id* with the OSPFv3 area identifier.
Step 6	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-ospfv3-ar)# interface gigabitEthernet 0/1/5/0`	Enters interface configuration mode and specifies the interface name and notation *rack/slot/module/port*. • The example indicates a Gigabit Ethernet interface in modular services card slot 1.
Step 7	**bfd fast-detect** **Example:** `RP/0/RP0/CPU0:router(config-ospfv3-ar-if)# bfd fast-detect`	Enables BFD to detect failures in the path between adjacent forwarding engines.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 9	**show run router ospfv3** **Example:** `RP/0/RP0/CPU0:router(config-ospfv3-ar-if)#show run router ospfv3`	Verifies that BFD is enabled on the appropriate interface.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
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.

	Command or Action	Purpose
Step 3	**bfd address-family ipv4 destination** *ip-address* **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 *ip-address* is the 32-bit IP address in dotted-decimal format (A.B.C.D).
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
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 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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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-id*
3. **bundle minimum-active bandwidth** *kbps*
4. **bundle minimum-active links** *links*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

	Command or Action	Purpose
Step 2	**interface Bundle-Ether** *bundle-id* **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** *kbps* **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** *links* **Example:** `RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2`	Sets the number of active links required before a bundle can be brought up or remain up. The range is from 1 to 32. **Note** • 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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
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` **Note** • 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 NCS 6000 Series Router is 33 ms.	
Step 4	**bfd address-family ipv4 multiplier** *multiplier* **Example:** `RP/0/RP0/CPU0:router(config-if)#bfd address-family ipv4 multiplier 30`	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. **Note** • Although the command allows you to configure a minimum of 2, the supported minimum is 3.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
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 timers start** *seconds* **Example:**	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

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-if)#`	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.
Step 4	**bfd address-family ipv4 timers nbr-unconfig** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-if)#`	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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Enabling 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.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**bfd** **Example:** `RP/0/RP0/CPU0:router(config)# bfd`	Enters BFD configuration mode.
Step 3	**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 *ip-address* is the 32-bit IP address in dotted-decimal format (A.B.C.D).
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
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. In BFD interface configuration mode, you can specify an IPv4 address on an individual interface.
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 *ip-address* is the 32-bit IP address in dotted-decimal format (A.B.C.D).
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BFD Session Teardown Based on Echo Latency Detection

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**bfd** **Example:** `RP/0/RP0/CPU0:router(config)# bfd`	Enters BFD configuration mode.
Step 3	**echo latency detect** [**percentage** *percent-value* [**count** *packet-count*] **Example:** `RP/0/RP0/CPU0:router(config-bfd)# echo latency detect`	Enables echo packet latency detection over the course of a BFD session, where: • **percentage** *percent-value*—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** *packet-count*—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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Delaying BFD Session Startup Until Verification of Echo Path and Latency

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**bfd** **Example:** `RP/0/0RP0RSP0/CPU0:router(config)# bfd`	Enters BFD configuration mode.
Step 3	**echo startup validate** [**force**] **Example:** `RP/0/0RP0RSP0/CPU0:router(config-bfd)# echo startup validate`	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 within the configured latency before allowing the BFD session to change state.

	Command or Action	Purpose
		When the **force** keyword is not configured, the local system performs echo startup validation if the following conditions are true: • 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). When the force keyword is configured, the local system performs echo startup validation if following conditions are true. • 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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Disabling 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.

Disabling Echo Mode on an Individual Interface

The following procedures describe how to disable echo mode on an interface .

SUMMARY STEPS

1. **configure**
2. **bfd**
3. **interface** *type interface-path-id*
4. **echo disable**
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
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 disable echo mode on an individual interface or bundle.
Step 4	**echo disable** **Example:** `RP/0/RP0/CPU0:router(config-bfd-if)# echo disable`	Disables echo mode on the specified individual interface or bundle.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**bfd** **Example:** `RP/0/RP0/CPU0:router(config)# bfd`	Enters BFD configuration mode.
Step 3	**dampening** [**bundle-member**] {**initial-wait** | **maximum-wait** | **secondary-wait**} *milliseconds* **Example:** `RP/0/RP0/CPU0:router(config-bfd)# dampening initial-wait 30000`	Specifies delays in milliseconds for BFD session startup to control flapping. The value for **maximum-wait** should be greater than the value for **initial-wait**. The dampening values can be defined for bundle member interfaces and for the non-bundle interfaces.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**bfd** **Example:** `RP/0/RP0/CPU0:router(config)# bfd`	Enters BFD configuration mode.
Step 3	**ipv6 checksum** [**disable**] **Example:**	Enables IPv6 checksum support on the interface. To disable, use the **disable** keyword.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-bfd-if)# ipv6 checksum disable`	
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**bfd** **Example:** `RP/0/RP0/CPU0:router(config)# bfd`	Enters BFD configuration mode.
Step 3	**interface** *type interface-path-id* **Example:**	Enters BFD interface configuration mode for a specific interface.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-bfd)# interface gigabitEthernet 0/1/5/0`	
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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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** *type interface-path-id***] location** *node-id* **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.
Step 2	**clear bfd counters** [**ipv4	ipv6	all**] **packet** [**interface** *type interface-path-id*] **location** *node-id* **Example:**	Clears the BFD counters for IPv4 packets, IPv6 packets, or all packets.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# clear bfd counters all packet location 0/3/cpu0`	
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 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**interface tunnel-te** *interface-number* **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 fast-detect** **Example:** `RP/0/RP0/CPU0:router(config-if)#bfd fast-detect`	Enables BFD fast detection.
Step 4	**bfd minimum-interval***milliseconds* **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** *number* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**interface tunnel-te** *interface-number* **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** *seconds* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**interface tunnel-te** *interface-number* **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 dampening initial-wait** *milliseconds* **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** *milliseconds* **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** *milliseconds* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**interface tunnel-te** *interface-number* **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	Use one of these commands: • **bfd lsp-ping interval** *300* **Example:** `RP/0/RP0/CPU0:router(config-if)#bfd lsp-ping interval 300` Or	Sets periodic interval for LSP ping requests or disables LSP ping requests. • **interval** *seconds*—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.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-if)#bfd lsp-ping disable`	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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**mpls traffic-eng bfd lsp tailminimum-interval** *milliseconds*	Configures hello interval in milliseconds.

	Command or Action	Purpose
	Example: `RP/0/RP0/CPU0:router(config)#mpls traffic-eng bfd lsp tail minimum-interval 20000`	Hello interval range is 100 to 30000 milliseconds. Default hello interval is 100 milliseconds
Step 3	**mpls traffic-eng bfd lsp tailmultiplier** *number* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

	Command or Action	Purpose
Step 2	**bfd** **Example:** `RP/0/RP0/CPU0:router(config)# bfd`	Enters BFD configuration mode.
Step 3	**multipath include location** *node-id* **Example:** `RP/0/RP0/CPU0:router(config-bfd)# multipath include location 0/1/CPU0`	Configures BFD multiple path on specific line card. BFD over LSP sessions, both head-end and tail-end, will be hosted on line cards. BFD over LSP sessions, both 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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BFD Object Tracking:

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type bfdrtr rate** *tx-rate*
4. **debounce***debounce*
5. **interface** *if-name*
6. **destaddress** *dest_addr*
7. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**track** *track-name* **Example:**	Enters track configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config)# track track1`	• *track-name*—Specifies a name for the object to be tracked.
Step 3	**type bfdrtr rate** *tx-rate* **Example:** `RP/0/RP0/CPU0:router(config-track)# type bfdrtr rate 4`	tx_rate - time in msec at which the BFD should probe the remote entity
Step 4	**debounce***debounce* **Example:** `RP/0/RP0/CPU0:router(config-if)# debounce 10`	debounce - count of consecutive BFD probes whose status should match before BFD notifies OT
Step 5	**interface** *if-name* **Example:** `RP/0/RP0/CPU0:router(config-track-line-prot)# interface atm 0/2/0/0.1`	if_name - interface name on the source to be used by BFD to check the remote BFD status.
Step 6	**destaddress** *dest_addr* **Example:** `RP/0/RP0/CPU0:router(config-if)#destaddress 1.2.3.4`	dest_addr - IPV4 address of the remote BFD entity being tracked.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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:

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.

BFD on Bundled VLANs: Example

The following example shows how to configure BFD on bundled VLANs:

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

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



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
```

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

 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
!
```

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:routerr(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	*Routing Command Reference for Cisco NCS 6000 Series Rou*
Configuring QoS packet classification	

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

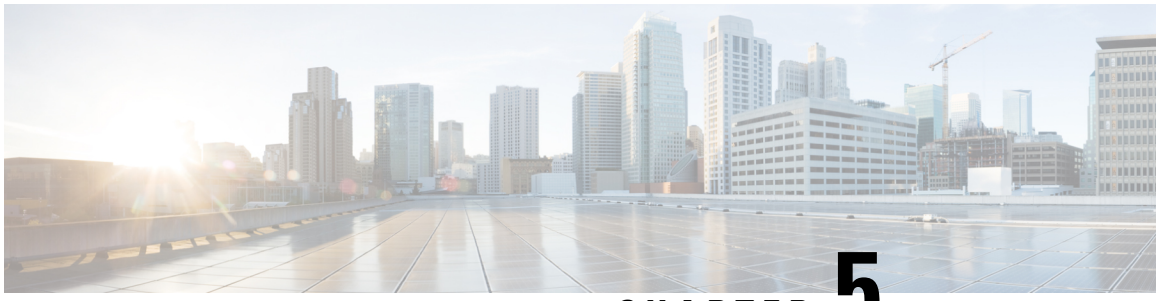
RFCs	Title
rfc5880_bfd_base	*Bidirectional Forwarding Detection*, June 2010
rfc5881_bfd_ipv4_ipv6	*BFD for IPv4 and IPv6 (Single Hop)*, June 2010
rfc5883_bfd_multihop	*BFD for Multihop Paths*, June 2010

MIBs

MIBs	MIBs Link
All	To locate and download MIBs using Cisco IOS XR software, Cisco MIB Locator tool found at the following URL and cho platform under the Cisco Access Products menu.

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.

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 NCS 6000 Series 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 5.0.0	This feature was introduced.

- Prerequisites for Implementing EIGRP, on page 209
- Restrictions for Implementing EIGRP , on page 210
- Information About Implementing EIGRP, on page 210
- How to Implement EIGRP , on page 219
- Configuration Examples for Implementing EIGRP , on page 230
- Additional References, on page 231

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.
- 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.

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. Some EIGRP packets must be sent reliably and others need not be. For efficiency, reliability is provided only when necessary.

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:

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)#
```

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)#
```

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.

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//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//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//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

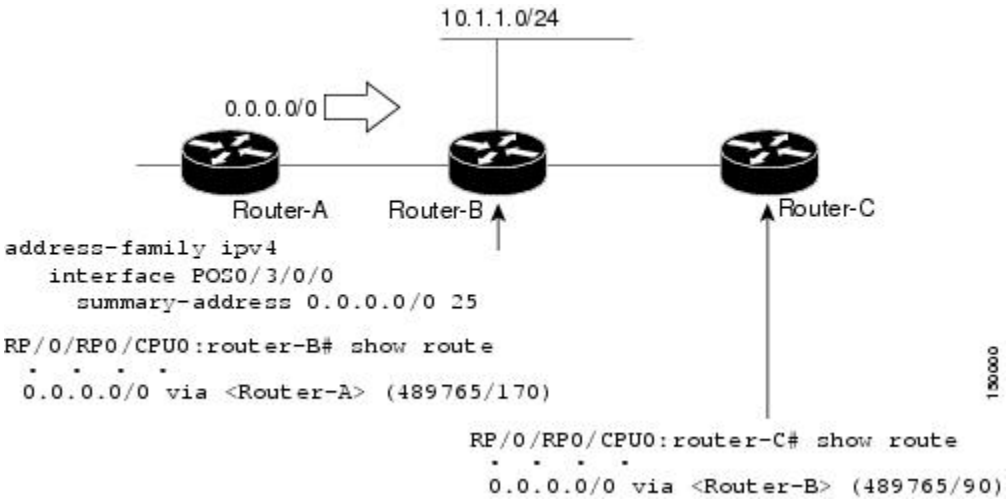
You can also use a floating summary route when configuring the **summary-address** command. The floating summary route is created by applying a default route and administrative distance at the interface level. The following scenario illustrates the behavior of this enhancement.

Figure 11: Floating Summary Route Is Applied to Router-B, on page 215 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 11: 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:

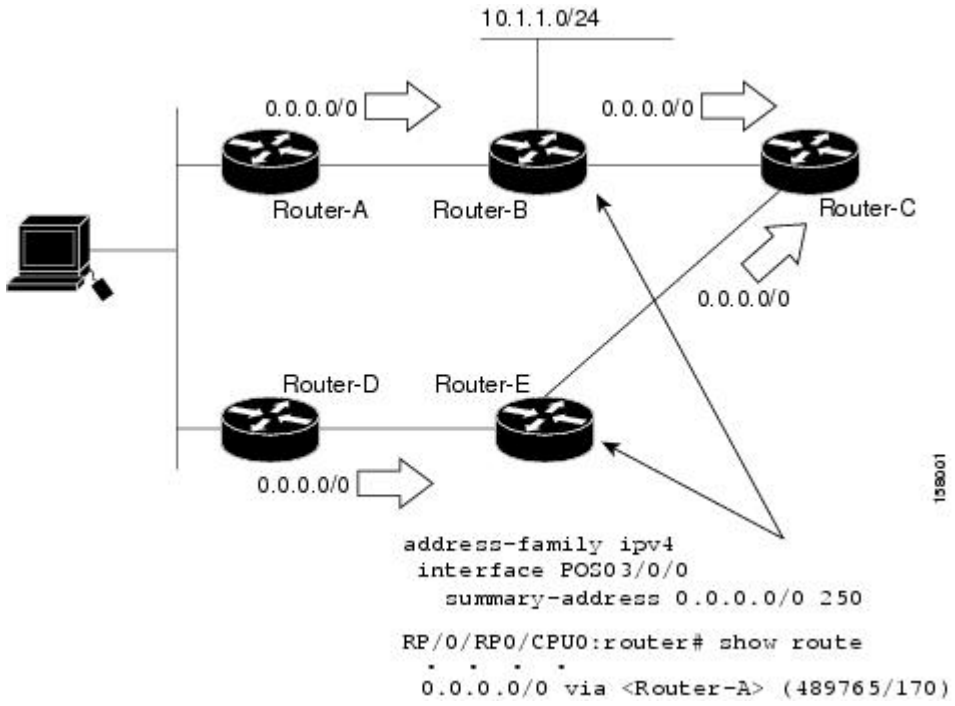
```
RP/0/RP0/CPU0:router(config-if)# summary-address 100 0.0.0.0 0.0.0.0 250
```

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 12: Floating Summary Route Applied for Dual-Homed Remotes, on page 216 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 12: 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.

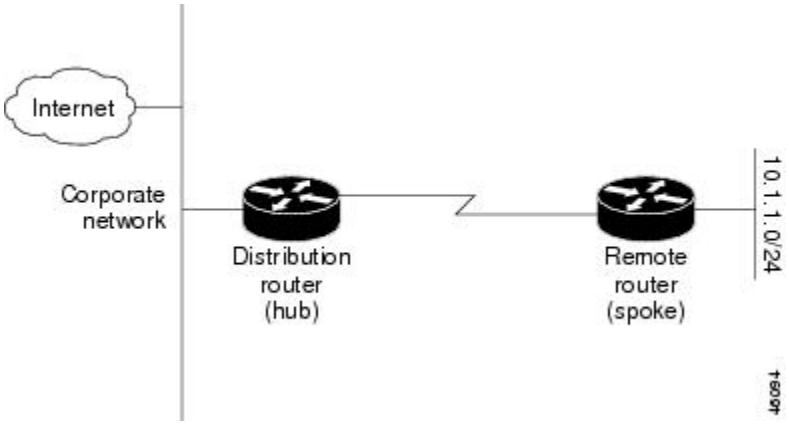
Stub routing is commonly used in a hub-and-spoke network topology. In a hub-and-spoke network, one or more end (stub) networks are connected to a remote router (the spoke) that is connected to one or more distribution routers (the hub). The remote router is adjacent only to one or more distribution routers. 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 13: 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 13: Simple Hub-and-Spoke Network, on page 217 , 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 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.

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** }
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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters XR Config mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# 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	**address-family** { **ipv4** } **Example:** `RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4`	Enters an address family configuration mode.
Step 4	**router-id** *id* **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.
Step 5	**default-metric** *bandwidth delay reliability loading mtu* **Example:** `RP/0/RP0/CPU0:router(config-eigrp-af)# default-metric 1000 100 250 100 1500`	(Optional) Sets metrics for an EIGRP process.
Step 6	**distance** *internal-distance external-distance* **Example:** `RP/0/RP0/CPU0:router(config-eigrp-af)# distance 80 130`	(Optional) Allows the use of two administrative distances—internal and external—that could be a better route to a node.
Step 7	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-eigrp-af)# interface GigabitEthernet 0/1/0/0`	Defines the interfaces on which the EIGRP routing protocol runs.
Step 8	**holdtime** seconds **Example:** `RP/0/RP0/CPU0:router(config-eigrp-af-if)# holdtime 30`	(Optional) Configures the hold time for an interface.
Step 9	**bandwidth-percent** *percent* **Example:**	(Optional) Configures the percentage of bandwidth that may be used by EIGRP on an interface.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-eigrp-af-if)# bandwidth-percent 75`	
Step 10	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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** }
4. **route-policy** *name* **out**
5. **interface** *type interface-path-id*
6. **summary-address** *ip-address* { **/** *length* | *mask* } [*admin-distance*]
7. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router eigrp** *as-number* **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** } **Example:** `RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4`	Enters an address family configuration mode.
Step 4	**route-policy** *name* **out** **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** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-eigrp-af)# interface GigabitEthernet 0/1/0/0`	Defines the interfaces on which the EIGRP routing protocol runs.
Step 6	**summary-address** *ip-address* { / *length* | *mask* } [*admin-distance*] **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router eigrp** *as-number* **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.
Step 4	**redistribute** {{ **bgp**	**connected**	**isis**	**ospf**	**ospfv3**	**rip**	**static** } [*as-number*]} [**route-policy** *name*] **Example:** `RP/0/RP0/CPU0:router(config-eigrp-af)# redistribute bgp 100`	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	**redistribute maximum-prefix** *maximum* [*threshold*] [[**dampened**] [**reset-time** *minutes*] [**restart** *minutes*] [**restart-count** *number*]	[**warning-only**]]	Limits the maximum number of prefixes that are redistributed to the EIGRP process.

	Command or Action	Purpose
	Example: `RP/0/RP0/CPU0:router(config-eigrp-af)# redistribute maximum-prefix 5000 95 warning-only`	**Caution** After the restart count threshold is crossed, you need to use the `clear eigrp neighbors` command to re-establish normal peering, redistribution, or both.
Step 6	**timers nsf route-hold** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-eigrp-af)# timers nsf route-hold 120`	Sets the timer that determines how long an NSF-aware EIGRP router holds routes for an inactive peer.
Step 7	**maximum paths** *maximum* **Example:** `RP/0/RP0/CPU0:router(config-eigrp-af)# maximum paths 10`	Controls the maximum number of parallel routes that the EIGRP can support.
Step 8	**maximum-prefix** *maximum* [*threshold*] [[**dampened**] [**reset-time** *minutes*] [**restart** *minutes*] [**restart-count** *number*] | [**warning-only]]** **Example:** `RP/0/RP0/CPU0:router(config-eigrp-af)# maximum-prefix 50000`	Limits the number of prefixes that are accepted under an address family by EIGRP.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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. Use the **commit** or **end** command.
6. **configure**
7. **router eigrp** *as-number*
8. **address-family** { **ipv4** | **ipv6** }
9. **route-policy** *route-policy-name* { **in** | **out** }
10. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**route-policy** *name* **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** *bandwidth delay reliability load mtu* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

	Command or Action	Purpose
Step 6	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 7	**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 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** *route-policy-name* { **in** | **out** } **Example:** `RP/0/RP0/CPU0:router(config-eigrp-af)# route-policy IN-IPv4 in`	Applies a routing policy to updates advertised to or received from an EIGRP neighbor.
Step 10	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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** }
4. **stub** [**receive-only** | {[**connected**] [**redistributed**] [**static**] [**summary**]}]
5. Use the **commit** or **end** command.
6. **show eigrp** [**ipv4**] **neighbors** [*as-number*] [**detail**] [*type interface-path-id* | **static**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
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	**address-family** { **ipv4** } **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:** `RP/0/RP0/CPU0:router(config-eigrp-af)# stub receive-only`	Configures a router as a stub for EIGRP.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 6	**show eigrp** [**ipv4**] **neighbors** [*as-number*] [**detail**] [*type interface-path-id* | **static**]	Verifies that a remote router has been configured as a stub router with EIGRP.

	Command or Action	Purpose
	Example: `RP/0/RP0/CPU0:router# show eigrp neighbors detail`	The last line of the output shows the stub status of the remote or spoke router.

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

1. **configure**
2. **router eigrp** *as-number*
3. **address-family** [**ipv4** | **ipv6**]
4. **log-neighbor-changes**
5. **log-neighbor-warnings**
6. Use the **commit** or **end** command.
7. **clear eigrp** [*as-number*] [**ipv4** | **ipv6**] **neighbors** [*ip-address* | *type interface-path-id*]
8. **clear eigrp** [*as-number*] [**ipv4** | **ipv6**] **topology** [*prefix mask*] [*prefix / length*]
9. **show eigrp** [*as-number*] [**ipv4** | **ipv6**] **interfaces** [*type interface-path-id*] [**detail**]
10. **show eigrp** [*as-number*] [**ipv4** | **ipv6**] **neighbors** [**detail**] [*type interface-path-id* | **static**]
11. **show protocols eigrp**
12. **show eigrp** [*as-number*] [**ipv4** | **ipv6**] **topology** [*ip-address mask*] [**active** | **all-links** | **detail-links** | **pending** | **summary** | **zero-successors**]
13. **show eigrp** [*as-number*] [**ipv4** | **ipv6**] **traffic**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
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	**address-family** [**ipv4**	**ipv6**] **Example:** `RP/0/RP0/CPU0:router(config-eigrp)# address-family ipv4`	Enters an address family configuration mode.

	Command or Action	Purpose
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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 7	**clear eigrp** [*as-number*] [**ipv4** | **ipv6**] **neighbors** [*ip-address* | *type interface-path-id*] **Example:** `RP/0/RP0/CPU0:routerr# clear eigrp 20 neighbors 0/1/0/0`	Deletes EIGRP neighbor entries from the appropriate table.
Step 8	**clear eigrp** [*as-number*] [**ipv4** | **ipv6**] **topology** [*prefix mask*] [*prefix / length*] **Example:** `RP/0/RP0/CPU0:router# clear eigrp topology`	Deletes EIGRP topology entries from the appropriate tab.
Step 9	**show eigrp** [*as-number*] [**ipv4** | **ipv6**] **interfaces** [*type interface-path-id*] [**detail**] **Example:** `RP/0/RP0/CPU0:router# show eigrp interfaces detail`	Displays information about interfaces configured for EIGRP.
Step 10	**show eigrp** [*as-number*] [**ipv4** | **ipv6**] **neighbors** [**detail**] [*type interface-path-id* | **static**] **Example:**	Displays the neighbors discovered by EIGRP.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# show eigrp neighbors 20 detail static`	
Step 11	**show protocols eigrp** **Example:** `RP/0/RP0/CPU0:router# show protocols eigrp`	Displays information about the EIGRP process configuration.
Step 12	**show eigrp** [*as-number*] [**ipv4** | **ipv6**] **topology** [*ip-address mask*] [**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 13	**show eigrp** [*as-number*] [**ipv4** | **ipv6**] **traffic** **Example:** `RP/0/RP0/CPU0:router# show eigrp traffic`	Displays the number of EIGRP packets sent and received.

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
      !
    !
  !
```

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	*Routing Command Reference for Cisco NCS 6000 Series Routers*
Site of Origin (SoO) support for EIGRP feature information	*Implementing MPLS Traffic Engineering on* module in *MPLS Configuration Guide for Cisco NCS 6000 Series Routers*

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

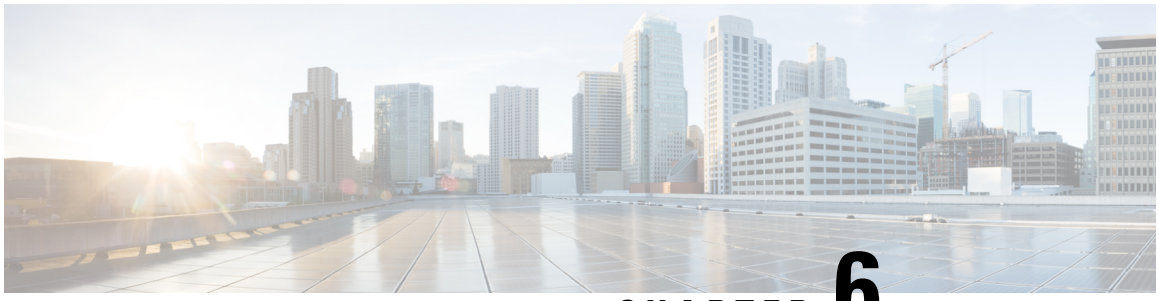
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

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 233
- Implementing IS-IS, on page 233
- Information About Implementing IS-IS , on page 233
- IS-IS Cost Fallback on IOS XR Bundle-Ether Interface, on page 292
- Configuration Examples for Implementing IS-IS , on page 294
- Where to Go Next, on page 296
- Additional References, on page 296

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.

Information About Implementing IS-IS

To implement IS-IS you need to understand the following concepts:

IS-IS Functional Overview

Small IS-IS networks are typically built as a single area that includes all routers in the network. As the network grows larger, it may be reorganized into a backbone area made up of the connected set of all Level 2 routers from all areas, which is in turn connected to local areas. Within a local area, routers know how to reach all system IDs. Between areas, routers know how to reach the backbone, and the backbone routers know how to reach other areas.

The IS-IS routing protocol supports the configuration of backbone Level 2 and Level 1 areas and the necessary support for moving routing information between the areas. Routers establish Level 1 adjacencies to perform routing within a local area (intra-area routing). Routers establish Level 2 adjacencies to perform routing between Level 1 areas (interarea routing).

Each IS-IS instance can support either a single Level 1 or Level 2 area, or one of each. By default, all IS-IS instances automatically support Level 1 and Level 2 routing. You can change the level of routing to be performed by a particular routing instance using the **is-type** command.

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.

Key Features Supported in the Cisco IOS XR IS-IS Implementation

The Cisco IOS XR implementation of IS-IS conforms to the IS-IS Version 2 specifications detailed in RFC 1195 and the IPv6 IS-IS functionality based on the Internet Engineering Task Force (IETF) IS-IS Working Group draft-ietf-isis-ipv6.txt document.

The following list outlines key features supported in the Cisco IOS XR implementation:

- Single topology IPv6
- Multitopology
- Nonstop forwarding (NSF), both Cisco proprietary and IETF
- Three-way handshake
- Mesh groups
- Multiple IS-IS instances
- Configuration of a broadcast medium connecting two networking devices as a point-to-point link
- Fast-flooding with different threads handling flooding and shortest path first (SPF).

Note For information on IS-IS support for Bidirectional Forwarding Detection (BFD), see *Interface and Hardware Component Configuration Guide for Cisco NCS 6000 Series Routers* and *Interface and Hardware Component Command Reference for the Cisco NCS 6000 Series Routers*.

IS-IS Configuration Grouping

Cisco IOS XR groups all of the IS-IS configuration in router IS-IS configuration mode, including the portion of the interface configurations associated with IS-IS. To display the IS-IS configuration in its entirety, use the **show running router isis** command. The command output displays the running configuration for all configured IS-IS instances, including the interface assignments and interface attributes.

IS-IS Configuration Modes

The following sections 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:

```
RP/0/RP0/CPU0:router# configuration
RP/0/RP0/CPU0:router(config)# router isis isp
RP/0/RP0/CPU0:router(config-isis)#
```

Router Address Family Configuration Mode

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

```
RP/0/RP0/CPU0:router(config)# router isis isp
RP/0/RP0/CPU0:router(config-isis)# address-family
ipv4 u
nicast
RP/0/RP0/CPU0:router(config-isis-af)#
```

Interface Configuration Mode

The following example shows how to enter interface configuration mode:

```
RP/0/RP0/CPU0:router(config)# router isis isp
RP/0/RP0/CPU0:router(config-isis)# interface  GigabitEthernet 0
/3/0/0
RP/0/RP0/CPU0:router(config-isis-if)#
```

Interface Address Family Configuration Mode

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

```
RP/0/RP0/CPU0:router(config)# router isis isp
RP/0/RP0/CPU0:router(config-isis)# interface
GigabitEthernet 0 /3/0/0
RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)#
```

IS-IS Interfaces

IS-IS interfaces can be configured as one of the following types:

- Active—advertises connected prefixes and forms adjacencies. This is the default 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 IS-IS domain. If many connected prefixes need to be advertised then the redistribution of connected routes with the appropriate policy should be used instead.
- Suppressed—does not advertise connected prefixes but forms adjacencies. The **suppress** command is used to configure interfaces as suppressed.
- Shutdown—does not advertise connected prefixes and does not form adjacencies. The **shutdown** command is used to disable interfaces without removing the IS-IS configuration.

Multitopology Configuration

The software supports multitopology for IPv6 IS-IS unless single topology is explicitly configured in IPv6 address-family configuration mode.

Note IS-IS supports IP routing and not Open Systems Interconnection (OSI) Connectionless Network Service (CLNS) routing.

IPv6 Routing and Configuring IPv6 Addressing

By default, IPv6 routing is disabled in the software. To enable IPv6 routing, you must assign IPv6 addresses to individual interfaces in the router using the **ipv6 enable** or **ipv6 address** command. See the Network Stack IPv4 and IPv6 Commands on module of *IP Addresses and Services Command Reference for Cisco NCS 6000 Series Routers*.

Limit LSP Flooding

Limiting link-state packets (LSP) may be desirable in certain “meshy” network topologies. An example of such a network might be a highly redundant one such as a fully meshed set of point-to-point links over a nonbroadcast multiaccess (NBMA) transport. In such networks, full LSP flooding can limit network scalability. One way to restrict the size of the flooding domain is to introduce hierarchy by using multiple Level 1 areas and a Level 2 area. However, two other techniques can be used instead of or with hierarchy: Block flooding on specific interfaces and configure mesh groups.

Both techniques operate by restricting the flooding of LSPs in some fashion. A direct consequence is that although scalability of the network is improved, the reliability of the network (in the face of failures) is reduced because a series of failures may prevent LSPs from being flooded throughout the network, even though links exist that would allow flooding if blocking or mesh groups had not restricted their use. In such a case, the link-state databases of different routers in the network may no longer be synchronized. Consequences such as persistent forwarding loops can ensue. For this reason, we recommend that blocking or mesh groups be used only if specifically required, and then only after careful network design.

Flood Blocking on Specific Interfaces

With this technique, certain interfaces are blocked from being used for flooding LSPs, but the remaining interfaces operate normally for flooding. This technique is simple to understand and configure, but may be more difficult to maintain and more error prone than mesh groups in the long run. The flooding topology that IS-IS uses is fine-tuned rather than restricted. Restricting the topology too much (blocking too many interfaces) makes the network unreliable in the face of failures. Restricting the topology too little (blocking too few interfaces) may fail to achieve the desired scalability.

To improve the robustness of the network in the event that all nonblocked interfaces drop, use the **csnp-interval** command in interface configuration mode to force periodic complete sequence number PDUs (CSNPs) packets to be used on blocked point-to-point links. The use of periodic CSNPs enables the network to become synchronized.

Mesh Group Configuration

Configuring mesh groups (a set of interfaces on a router) can help to limit flooding. All routers reachable over the interfaces in a particular mesh group are assumed to be densely connected with each router having at least one link to every other router. Many links can fail without isolating one or more routers from the network.

In normal flooding, a new LSP is received on an interface and is flooded out over all other interfaces on the router. With mesh groups, when a new LSP is received over an interface that is part of a mesh group, the new LSP is not flooded over the other interfaces that are part of that mesh group.

Maximum LSP Lifetime and Refresh Interval

By default, the router sends a periodic LSP refresh every 15 minutes. LSPs remain in a database for 20 minutes by default. If they are not refreshed by that time, they are deleted. You can change the LSP refresh interval or maximum LSP lifetime. The LSP interval should be less than the LSP lifetime or else LSPs time out before they are refreshed. In the absence of a configured refresh interval, the software adjusts the LSP refresh interval, if necessary, to prevent the LSPs from timing out.

Single-Topology IPv6 Support

Single-topology IPv6 support on Cisco IOS XR software software allows IS-IS for IPv6 to be configured on interfaces along with an IPv4 network protocol. All interfaces must be configured with the identical set of network protocols, and all routers in the IS-IS area (for Level 1 routing) or the domain (for Level 2 routing) must support the identical set of network layer protocols on all interfaces.

In single-topology mode, IPv6 topologies work with both narrow and wide metric styles in IPv4 unicast topology. During single-topology operation, one shortest path first (SPF) computation for each level is used to compute both IPv4 and IPv6 routes. Using a single SPF is possible because both IPv4 IS-IS and IPv6 IS-IS routing protocols share a common link topology.

Multitopology IPv6 for IS-IS

Multitopology IPv6 for IS-IS assumes that multitopology support is required as soon as it detects interfaces configured for both IPv6 and IPv4 within the IS-IS stanza.

Because multitopology is the default behavior in the software, you must explicitly configure IPv6 to use the same topology as IPv4 to enable single-topology IPv6. Configure the **single-topology** command in IPv6 router address family configuration submode of the IS-IS router stanza.

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
```

IS-IS Authentication

Authentication is available to limit the establishment of adjacencies by using the **hello-password** command, and to limit the exchange of LSPs by using the **lsp-password** command.

IS-IS supports plain-text authentication, which does not provide security against unauthorized users. Plain-text authentication allows you to configure a password to prevent unauthorized networking devices from forming adjacencies with the router. The password is exchanged as plain text and is potentially visible to an agent able to view the IS-IS packets.

When an HMAC-MD5 password is configured, the password is never sent over the network and is instead used to calculate a cryptographic checksum to ensure the integrity of the exchanged data.

IS-IS stores a configured password using simple encryption. However, the plain-text form of the password is used in LSPs, sequence number protocols (SNPs), and hello packets, which would be visible to a process that can view IS-IS packets. The passwords can be entered in plain text (clear) or encrypted form.

To set the domain password, configure the **lsp-password** command for Level 2; to set the area password, configure the **lsp-password** command for Level 1.

The keychain feature allows IS-IS to reference configured keychains. IS-IS key chains enable hello and LSP keychain authentication. Keychains can be configured at the router level (in the case of the **lsp-password** command) and at the interface level (in the case of the **hello-password** command) within IS-IS. These commands reference the global keychain configuration and instruct the IS-IS protocol to obtain security parameters from the global set of configured keychains.

IS-IS is able to use the keychain to implement hitless key rollover for authentication. ey 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).

Nonstop Forwarding

On Cisco IOS XR software, IS-IS NSF minimizes the amount of time a network is unavailable to its users following the restart of the IS-IS process.

When the IS-IS process restarts, all routing peers of that device usually detect that the device went down and then came back up. This transition results in what is called a *routing flap*, which could spread across multiple routing domains. Routing flaps caused by routing restarts create routing instabilities, which are detrimental to the overall network performance. NSF helps to suppress routing flaps, thus reducing network instability.

NSF allows for the forwarding of data packets to continue along known routes while the routing protocol information is being restored following the process restarts. When the NSF feature is configured, peer networking devices do not experience routing flaps. To preserve routing across RP failover events, NSR must be configured in addition to NSF.

When the Cisco IOS XR router running IS-IS routing performs the process restarts, the router must perform two tasks to resynchronize its link-state database with that of its IS-IS neighbors. First, it must relearn the available IS-IS 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.

The IS-IS NSF feature offers two options when configuring NSF:

- IETF NSF
- Cisco NSF

If neighbor routers on a network segment are NSF-aware, meaning that they are running a software version that supports RFC5306, they assist a router configured with **nsf ietf** command that is restarting. IETF NSF enables the neighbor routers provide adjacency and link-state information to help rebuild the routing information following a failover.

In Cisco IOS XR software, Cisco NSF checkpoints (stores persistently) all the state necessary to recover from a restart without requiring any special cooperation from neighboring routers. The state is recovered from the neighboring routers, but only using the standard features of the IS-IS routing protocol. This capability makes Cisco NSF suitable for use in networks in which other routers have not used the IETF standard implementation of NSF.

Note If you configure IETF NSF on the Cisco IOS XR router and a neighbor router does not support IETF NSF, the affected adjacencies flap, but nonstop forwarding is maintained to all neighbors that do support IETF NSF. A restart reverts to a cold start if no neighbors support IETF NSF.

Note ISIS, as a routing protocol, supports shorter hello intervals for neighbor adjacency management. However, the nonstop forwarding or routing feature on Cisco routers only work for default hello interval. When a failover occurs, there is a delay before the hello messages are sent and before the system is ready to send or receive the packets. The nonstop forwarding or routing feature is supported with a default hello interval or multiplier to overcome this delay. BFD, which runs on line cards, is responsible to fast-detect the loss of connectivity so there is no need to shorten hello interval.

Note Currently, a user can configure an aggressive hello-interval (lower than the default of 10 seconds for peer-to-peer session). But, if NSF is configured as a recovery for RP switchover, the default hello interval has to be used so that the sessions do not run into the risk of flapping during switchover.

Using LAN adjacencies in high availability (HA) scenarios is not recommended, since there is no designated intermediate system (DIS) redundancy in the protocol and traffic will either drop or be rerouted temporarily during DIS re-election.

ISIS NSR

Non Stop Routing (NSR) suppresses IS-IS routing changes for devices with redundant route processors during processor switchover events (RP failover or ISSU), reducing network instability and downtime. When Non Stop Routing is used, switching from the active to standby RP have no impact on the other IS-IS routers in the network. All information needed to continue the routing protocol peering state is transferred to the standby processor prior to the switchover, so it can continue immediately upon a switchover.

To preserve routing across process restarts, NSF must be configured in addition to NSR.

Configuring IS-IS Adjacency Stagger

Certain events like process restart or reload can involve a significant processing overhead. Updating routing tables with all adjacencies, maintaining them, and synchronizing the database with each adjacent router requires a lot of bandwidth. These processes may require large number of packets being sent and/or received, depending on the state of the database on the routers. If packets are dropped in any direction, it can lead to an unstable state.

We cannot prevent events like process restart or reload, but we can handle such events better by limiting the number of adjacencies that area being established simultaneously. To limit the number of adjacencies from getting established simultaneously, you can configure adjacency stagger. By configuring IS-IS adjacency stagger, you can specify the initial number neighbourhood routers from which adjacencies can fully form after a process restart or reload. If you configure IS-IS adjacency stagger, you can also specify the subsequent number of simultaneous neighbors that are allowed to form adjacency.

Restrictions

- IS-IS adjacency stagger is only supported on point-to-point interfaces and not on LAN interfaces.
- IS-IS adjacency stagger is not supported with NSF (non-stop forwarding) mechanisms.

Configuration Example

To configure IS-IS adjacency stagger on a point-to-point interface, you must use the following configuration steps:

1. Configure IS-IS.
2. Configure adjacency stagger.

Configuration

```
/* Enter the global configuration mode and configure IS-IS */
Router# config
Router(config)# router isis 1

/* Configure IS-IS adjacency stagger */
Router(config-isis)# adjacency stagger 2 3
Router(config-isis)# commit
```

Multi-Instance IS-IS

You can configure up to 16 IS-IS instances. MPLS can run on multiple IS-IS processes as long as the processes run on different sets of interfaces. Each interface may be associated with only a single IS-IS instance. Cisco IOS XR software prevents the double-booking of an interface by two instances at configuration time; two instances of MPLS configuration cause an error.

Because the Routing Information Base (RIB) treats each of the IS-IS instances as equal routing clients, you must be careful when redistributing routes between IS-IS instances. The RIB does not know to prefer Level 1 routes over Level 2 routes. For this reason, if you are running Level 1 and Level 2 instances, you must enforce the preference by configuring different administrative distances for the two instances.

Multiprotocol Label Switching Traffic Engineering

The MPLS TE feature enables an MPLS backbone to replicate and expand the traffic engineering capabilities of Layer 2 ATM and Frame Relay networks. MPLS is an integration of Layer 2 and Layer 3 technologies.

For IS-IS, MPLS TE automatically establishes and maintains MPLS TE label-switched paths across the backbone by using Resource Reservation Protocol (RSVP). The route that a label-switched path uses is determined by the label-switched paths resource requirements and network resources, such as bandwidth. Available resources are flooded by using special IS-IS TLV extensions in the IS-IS. The label-switched paths are explicit routes and are referred to as traffic engineering (TE) tunnels.

Overload Bit on Router

The overload bit is a special bit of state information that is included in an LSP of the router. If the bit is set on the router, it notifies routers in the area that the router is not available for transit traffic. This capability is useful in four situations:

1. During a serious but nonfatal error, such as limited memory.
2. During the startup and restart of the process. The overload bit can be set until the routing protocol has converged. However, it is not employed during a normal NSF restart or failover because doing so causes a routing flap.
3. During a trial deployment of a new router. The overload bit can be set until deployment is verified, then cleared.
4. During the shutdown of a router. The overload bit can be set to remove the router from the topology before the router is removed from service.

Overload Bit Configuration During Multitopology Operation

Because the overload bit applies to forwarding for a single topology, it may be configured and cleared independently for IPv4 and IPv6 during multitopology operation. For this reason, the overload is set from the router address family configuration mode. If the IPv4 overload bit is set, all routers in the area do not use the router for IPv4 transit traffic. However, they can still use the router for IPv6 transit traffic.

IS-IS Overload Bit Avoidance

The IS-IS overload bit avoidance feature allows network administrators to prevent label switched paths (LSPs) from being disabled when a router in that path has its Intermediate System-to-Intermediate System (IS-IS) overload bit set.

When the IS-IS overload bit avoidance feature is activated, all nodes with the overload bit set, including head nodes, mid nodes, and tail nodes, are ignored, which means that they are still available for use with label switched paths (LSPs).

Note The IS-IS overload bit avoidance feature does *not* change the default behavior on nodes that have their overload bit set if those nodes are not included in the path calculation (PCALC).

The IS-IS overload bit avoidance feature is activated using the following command:

mpls traffic-eng path-selection ignore overload

The IS-IS overload bit avoidance feature is deactivated using the **no** form of this command:

no mpls traffic-eng path-selection ignore overload

When the IS-IS overload bit avoidance feature is deactivated, nodes with the overload bit set cannot be used as nodes of last resort.

Default Routes

You can force a default route into an IS-IS routing domain. Whenever you specifically configure redistribution of routes into an IS-IS routing domain, the Cisco IOS XR software does not, by default, redistribute the default route into the IS-IS routing domain. The **default-information originate** command generates a *default route* into IS-IS, which can be controlled by a route policy. You can use the route policy to identify the level into which the default route is to be announced, and you can specify other filtering options configurable under a route policy. You can use a route policy to conditionally advertise the default route, depending on the existence of another route in the routing table of the router.

Attached Bit on an IS-IS Instance

The attached bit is set in a router that is configured with the **is-type** command and **level-1-2** keyword. The attached bit indicates that the router is connected to other areas (typically through the backbone). This functionality means that the router can be used by Level 1 routers in the area as the default route to the backbone. The attached bit is usually set automatically as the router discovers other areas while computing its Level 2 SPF route. The bit is automatically cleared when the router becomes detached from the backbone.

Note If the connectivity for the Level 2 instance is lost, the attached bit in the Level 1 instance LSP would continue sending traffic to the Level 2 instance and cause the traffic to be dropped.

To simulate this behavior when using multiple processes to represent the **level-1-2** keyword functionality, you would manually configure the attached bit on the Level 1 process.

IS-IS Support for Route Tags

The IS-IS Support for route tags feature provides the capability to associate and advertise a tag with an IS-IS route prefix. Additionally, the feature allows you to prioritize the order of installation of route prefixes in the RIB based on a tag of a route. Route tags may also be used in route policy to match route prefixes (for example, to select certain route prefixes for redistribution).

MPLS Label Distribution Protocol IGP Synchronization

Multiprotocol Label Switching (MPLS) Label Distribution Protocol (LDP) Interior Gateway Protocol (IGP) Synchronization ensures that LDP has completed label exchange before the IGP path is used for switching. MPLS traffic loss can occur in the following two situations:

- When an IGP adjacency is established, the router begins forwarding packets using the new adjacency before LDP has exchanged labels with peers on that link.
- When an LDP session closes, the router continues to forward traffic using the link associated with the LDP peer rather than using an alternate path with an established LDP session.

This feature provides a mechanism to synchronize LDP and IS-IS to minimize MPLS packet loss. The synchronization is accomplished by changing the link metric for a neighbor IS-IS link-state packet (LSP), based on the state of the LDP session.

When an IS-IS adjacency is established on a link but the LDP session is lost or LDP has not yet completed exchanging labels, IS-IS advertises the maximum metric on that link. In this instance, LDP IS-IS synchronization is not yet achieved.

Note In IS-IS, a link with a maximum wide metric (0xFFFFFF) is not considered for shortest path first (SPF). Therefore, the maximum wide metric of -1 (0XFFFFFE) is used with MPLS LDP IGP synchronization.

When LDP IS-IS synchronization is achieved, IS-IS advertises a regular (configured or default) metric on that link.

MPLS LDP-IGP Synchronization Compatibility with LDP Graceful Restart

LDP graceful restart protects traffic when an LDP session is lost. If a graceful restart-enabled LDP session fails, MPLS LDP IS-IS synchronization is still achieved on the interface while it is protected by graceful restart. MPLS LDP IGP synchronization is eventually lost under the following circumstances:

- LDP fails to restart before the LDP graceful restart reconnect timer expires.
- The LDP session on the protected interface fails to recover before the LDP graceful restart recovery timer expires.

MPLS LDP-IGP Synchronization Compatibility with IGP Nonstop Forwarding

IS-IS nonstop forwarding (NSF) protects traffic during IS-IS process restarts and route processor (RP) failovers. LDP IS-IS synchronization is supported with IS-IS NSF only if LDP graceful restart is also enabled over the interface. If IS-IS NSF is not enabled, the LDP synchronization state is not retained across restarts and failovers.

Label Distribution Protocol IGP Auto-configuration

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. LDP IGP auto-configuration can be used on a large number interfaces (for example, when LDP is used for transport in the core) and on multiple IGP instances simultaneously.

This feature supports the IPv4 address family for the default VPN routing and forwarding (VRF) instance.

LDP IGP auto-configuration can also be explicitly disabled on individual interfaces under LDP using the **igp auto-config disable** command. This allows LDP to receive all IGP interfaces except the ones explicitly disabled.

See the MPLS configuration guide for information on configuring LDP IGP auto-configuration.

MPLS TE Forwarding Adjacency

MPLS TE forwarding adjacency allows a network administrator to handle a traffic engineering, label switch path (LSP) tunnel as a link in an Interior Gateway Protocol (IGP) network, based on the Shortest Path First (SPF) algorithm. A forwarding adjacency can be created between routers in the same IS-IS level. The routers can be located multiple hops from each other. As a result, a TE tunnel is advertised as a link in an IGP network, with the cost of the link associated with it. Routers outside of the TE domain see the TE tunnel and use it to compute the shortest path for routing traffic throughout the network.

MPLS TE forwarding adjacency is considered in IS-IS SPF only if a two-way connectivity check is achieved. This is possible if the forwarding adjacency is bidirectional or the head end and tail end routers of the MPLS TE tunnel are adjacent.

The MPLS TE forwarding adjacency feature is supported by IS-IS. For details on configuring MPLS TE forwarding adjacency, see the MPLS Configuration Guide.

MPLS TE Interarea Tunnels

MPLS TE interarea tunnels allow you to establish MPLS TE tunnels that span multiple IGP areas (Open Shorted Path First [OSPF]) and levels (IS-IS), removing the restriction that required that both the tunnel headend and tailend routers be in the same area. The IGP can be either IS-IS or OSPF. See the Configuring MPLS Traffic Engineering for IS-IS, on page 263 for information on configuring MPLS TE for IS-IS.

For details on configuring MPLS TE interarea tunnels, see the MPLS Configuration Guide.

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. See IETF draft-ietf-rtgwg-ipfrr-framework-06.txt and draft-ietf-rtgwg-lf-conv-frmwk-00.txt for detailed information on IPFRR LFA.

IPFRR LFA is different from Multiprotocol Label Switching (MPLS) as it is applicable to networks using conventional IP routing and forwarding. See *MPLS Configuration Guide for Cisco NCS 6000 Series Routers* for information on configuring MPLS IPFRR.

IS-IS Over GRE Interfaces

Cisco IOS XR software provides the capability to run IS-IS protocols over Generic Routing Encapsulation (GRE) tunnel interfaces.

For more information on GRE tunnel interfaces, see *Implementing BGP on Cisco IOS XR software module.*

Unequal Cost Multipath Load-balancing for IS-IS

The unequal cost multipath (UCMP) load-balancing adds the capability with intermediate system-to-intermediate system (IS-IS) to load-balance traffic proportionally across multiple paths, with different cost.

Generally, higher bandwidth paths 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 paths or higher cost paths for traffic, and can install these paths to the forwarding information base (FIB). IS-IS IGP still 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 IS-IS per address family, enabling UCMP computation for a particular address family. 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 IS-IS. The number of UCMP nexthops 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) 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.

Enabling the UCMP configuration indicates that IS-IS should perform UCMP computation for the all the reachable ISIS prefixes or all the prefixes in 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 delay of ISIS_UCMP_INITIAL_DELAY (default delay is 100 ms) milliseconds from the time route calculation is completed and UCMP computation is started. UCMP computation will be done before fast re-route computation. Fast re-route backup paths will be calculated for both the primary equal cost multipath (ECMP) paths and the UCMP paths. Use the **ucmp delay-interval** command to configure the delay between primary SPF completion and start of UCMP computation.

To manually change each path's bandwidth to adjust UCMP ratio, use the **bandwidth**command in interface configuration mode.

UCMP ratio can be adjusted by any of the following ways:

- By using the **bandwidth** command in interface configuration mode to manually change the UCMP ratio.
- By adjusting the ISIS metric on the links.

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.

More than 32 ECMP and UCMP paths are not supported for these features:

- LI
- GRE

- BVI
- NetFlow
- Satellite
- MCAST
- SPAN
- PWHE
- ABF
- P2MP
- MVPN
- VPLS
- L2TPv3
- LISP
- VIDMON
- PBR

During a route processor failover (RPFO), IS-IS does not maintain maximum paths for a certain minimum amount of time. During this time, there is a minor amount of traffic flowing through a certain number of paths beyond the configured number of maximum paths.

To avoid such a scenario in Cisco access points, the interfaces are configured to have a maximum number of maximum paths so that traffic is forced to choose one among the configured ECMP maximum paths only.

Enabling IS-IS and Configuring Level 1 or Level 2 Routing

This task explains how to enable IS-IS and configure the routing level for an area.

Note Configuring the routing level in Step 4 is optional, but is highly recommended to establish the proper level of adjacencies.

Before you begin

Although you can configure IS-IS before you configure an IP address, no IS-IS routing occurs until at least one IP address is configured.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **net** *network-entity-title*
4. **is-type** { **level-1** | **level-1-2** | **level-2-only** }
5. Use the **commit** or **end** command.

6. **show isis** [**instance** *instance-id*] **protocol**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **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. • By default, all IS-IS instances are automatically Level 1 and Level 2. You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.
Step 3	**net** *network-entity-title* **Example:** `RP/0/RP0/CPU0:router(config-isis)# net 47.0004.004d.0001.0001.0c11.1110.00`	Configures network entity titles (NETs) for the routing instance. • Specify a NET for each routing instance if you are configuring multi-instance IS-IS. • This example configures a router with area ID 47.0004.004d.0001 and system ID 0001.0c11.1110.00. • To specify more than one area address, specify additional NETs. Although the area address portion of the NET differs, the systemID portion of the NET must match exactly for all of the configured items.
Step 4	**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. • The **level-1** keyword configures the software to perform Level 1 (intra-area) routing only. Only Level 1 adjacencies are established. The software learns about destinations inside its area only. Any packets containing destinations outside the area are sent to the nearest **level-1-2** router in the area. • The **level-2-only** keyword configures the software to perform Level 2 (backbone) routing only, and the router establishes only Level 2 adjacencies, 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

	Command or Action	Purpose
		router acts as a border router between the Level 2 backbone and its Level 1 area.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 6	**show isis** [**instance** *instance-id*] **protocol** **Example:** `RP/0/RP0/CPU0:router# show isis protocol`	(Optional) Displays summary information about the IS-IS instance.

Configuring Single Topology for IS-IS

After an IS-IS instance is enabled, it must be configured to compute routes for a specific network topology.

This task explains how to configure the operation of the IS-IS protocol on an interface for an IPv4 or IPv6 topology.

Before you begin

Note To enable the router to run in single-topology mode, configure each of the IS-IS interfaces with all of the address families enabled and "single-topology" in the address-family IPv6 unicast in the IS-IS router stanza. You can use either the IPv6 address family or both IPv4 and IPv6 address families, but your configuration must represent the set of all active address families on the router. Additionally, explicitly enable single-topology operation by configuring it in the IPv6 router address family submode.

Two exceptions to these instructions exist:

1. If the address-family stanza in the IS-IS process contains the **adjacency-check disable** command, then an interface is not required to have the address family enabled.
2. The **single-topology** command is not valid in the ipv4 address-family submode.

The default metric style for single topology is narrow metrics. However, you can use either wide metrics or narrow metrics. How to configure them depends on how single topology is configured. If both IPv4 and IPv6 are enabled and single topology is configured, the metric style is configured in the **address-family ipv4** stanza. You may configure the metric style in the **address-family ipv6** stanza, but it is ignored in this case. If only IPv6 is enabled and single topology is configured, then the metric style is configured in the **address-family ipv6** stanza.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. Do one of the following:
   - **ipv4 address** *address mask*
   - **ipv6 address** *ipv6-prefix / prefix-length* [**eui-64**]
   - **ipv6 address** *ipv6-address* { */ prefix-length* | *link-local* }
   - **ipv6 enable**
4. **exit**
5. **router isis** *instance-id*
6. **net** *network-entity-title*
7. **address-family ipv6** [**unicast**]
8. **single-topology**
9. exit
10. **interface** *type interface-path-id*
11. **circuit-type** { **level-1** | **level-1-2** | **level-2-only** }
12. **address-family** { **ipv4** | **ipv6** } [**unicast**]
13. Use the **commit** or **end** command.
14. **show isis** [**instance** *instance-id*] **interface** [*type interface-path-id*] [**detail**] [**level** { **1** | **2** }]
15. **show isis** [**instance** *instance-id*] **topology** [**systemid** *system-id*] [**level** { **1** | **2** }] [**summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface GigabitEthernet 0/1/0/3`	Enters interface configuration mode.

Step 3	Do one of the following:
	• **ipv4 address** *address mask*
	• **ipv6 address** *ipv6-prefix* / *prefix-length* [**eui-64**]
	• **ipv6 address** *ipv6-address* { / *prefix-length* | *link-local* }
	• **ipv6 enable**
	Example:
	`RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.0.1.3 255.255.255.0`
	or
	`RP/0/RP0/CPU0:router(config-if)# ipv6 address 3ffe:1234:c18:1::/64 eui-64`
	`RP/0/RP0/CPU0:router(config-if)# ipv6 address FE80::260:3EFF:FE11:6770 link-local`
	`RP/0/RP0/CPU0:router(config-if)# ipv6 enable`
	or
		Defines the IPv4 address for the interface. An IP address is required on all interfaces in an area enabled for IS-IS if any one interface is configured for IS-IS routing.
		or
		Specifies an IPv6 network assigned to the interface and enables IPv6 processing on the interface with the **eui-64** keyword.
		or
		Specifies an IPv6 address assigned to the interface and enables IPv6 processing on the interface with the **link-local** keyword.
		or
		Automatically configures an IPv6 link-local address on the interface while also enabling the interface for IPv6 processing.
		• The link-local address can be used only to communicate with nodes on the same link.
		• Specifying the **ipv6 address** *ipv6-prefix* / *prefix-length* interface configuration command without the **eui-64** keyword configures site-local and global IPv6 addresses.
		• Specifying the **ipv6 address** *ipv6-prefix* / *prefix-length* command with the **eui-64** keyword configures site-local and global IPv6 addresses with an interface ID in the low-order 64 bits of the IPv6 address. Only the 64-bit network prefix for the address needs to be specified; the last 64 bits are automatically computed from the interface ID.
		• Specifying the **ipv6 address** command with the **link-local** keyword configures a link-local address on the interface that is used instead of the link-local

	Command or Action	Purpose
		address that is automatically configured when IPv6 is enabled on the interface.
Step 4	**exit** **Example:** `RP/0/RP0/CPU0:router(config-if)# exit`	Exits interface configuration mode, and returns the router to XR Config mode.
Step 5	**router isis** *instance-id* **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. • By default, all IS-IS instances are Level 1 and Level 2. You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.
Step 6	**net** *network-entity-title* **Example:** `RP/0/RP0/CPU0:router(config-isis)# net 47.0004.004d.0001.0001.0c11.1110.00`	Configures NETs for the routing instance. • Specify a NET for each routing instance if you are configuring multi-instance IS-IS. You can specify a name for a NET and for an address. • This example configures a router with area ID 47.0004.004d.0001 and system ID 0001.0c11.1110.00. • To specify more than one area address, specify additional NETs. Although the area address portion of the NET differs, the system ID portion of the NET must match exactly for all of the configured items.
Step 7	**address-family ipv6** [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# address-family ipv6 unicast`	Specifies the IPv6 address family and enters router address family configuration mode. • This example specifies the unicast IPv6 address family.
Step 8	**single-topology** **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# single-topology`	(Optional) Configures the link topology for IPv4 when IPv6 is configured. • The **single-topology** command is valid only in IPv6 submode. The command instructs IPv6 to use the single topology rather than the default configuration of a separate topology in the multitopology mode. • See the Single-Topology IPv6 Support, on page 237 for more information.
Step 9	exit **Example:**	Exits router address family configuration mode, and returns the router to router configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-isis-af)# exit`	
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3`	Enters interface configuration mode.
Step 11	**circuit-type** { **level-1** | **level-1-2** | **level-2-only** } **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# circuit-type level-1-2`	(Optional) Configures the type of adjacency. • The default circuit type is the configured system type (configured through the **is-type** command). • Typically, the circuit type must be configured when the router is configured as only **level-1-2** and you want to constrain an interface to form only **level-1** or **level-2-only** adjacencies.
Step 12	**address-family** { **ipv4** | **ipv6** } [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters interface address family configuration mode. • This example specifies the unicast IPv4 address family on the interface.
Step 13	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 14	**show isis** [**instance** *instance-id*] **interface** [*type interface-path-id*] [**detail**] [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router# show isis interface GigabitEthernet 0/1/0/1`	(Optional) Displays information about the IS-IS interface.
Step 15	**show isis** [**instance** *instance-id*] **topology** [**systemid** *system-id*] [**level** { **1** | **2** }] [**summary**] **Example:** `RP/0/RP0/CPU0:router# show isis topology`	(Optional) Displays a list of connected routers in all areas.

Controlling LSP Flooding for IS-IS

Flooding of LSPs can limit network scalability. You can control LSP flooding by tuning your LSP database parameters on the router globally or on the interface. This task is optional.

Many of the commands to control LSP flooding contain an option to specify the level to which they apply. Without the option, the command applies to both levels. If an option is configured for one level, the other level continues to use the default value. To configure options for both levels, use the command twice. For example:

```
RP/0/RP0/CPU0:router(config-isis)# lsp-refresh-interval 1200 level 2
RP/0/RP0/CPU0:router(config-isis)# lsp-refresh-interval 1100 level 1
```

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **lsp-refresh-interval** *seconds* [**level** { **1** | **2** }]
4. **lsp-check-interval** *seconds* [**level** { **1** | **2** }]
5. **lsp-gen-interval** { [**initial-wait** *initial* | **secondary-wait** *secondary* | **maximum-wait** *maximum*] **. . .** } [**level** { **1** | **2** }]
6. **lsp-mtu** *bytes* [**level** { **1** | **2** }]
7. **max-lsp-lifetime** *seconds* [**level** { **1** | **2** }]
8. **ignore-lsp-errors disable**
9. **interface** *type interface-path-id*
10. **lsp-interval** *milliseconds* [**level** { **1** | **2** }]
11. **csnp-interval** *seconds* [**level** { **1** | **2** }]
12. **retransmit-interval** *seconds* [**level** { **1** | **2** }]
13. **retransmit-throttle-interval** *milliseconds* [**level** { **1** | **2** }]
14. **mesh-group** { *number* | **blocked** }
15. Use the **commit** or **end** command.
16. **show isis interface** [*type interface-path-id* | **level** { **1** | **2** }] [**brief**]
17. **show isis** [**instance** *instance-id*] **database** [**level** { **1** | **2** }] [**detail** | **summary** | **verbose**] [***** | *lsp-id*]
18. **show isis** [**instance** *instance-id*] **lsp-log** [**level** { **1** | **2** }]
19. **show isis database-log** [**level** { **1** | **2** }]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

	Command or Action	Purpose
Step 2	**router isis** *instance-id* **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. • You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.
Step 3	**lsp-refresh-interval** *seconds* [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis)# lsp-refresh-interval 10800`	(Optional) Sets the time between regeneration of LSPs that contain different sequence numbers • The refresh interval should always be set lower than the **max-lsp-lifetime** command.
Step 4	**lsp-check-interval** *seconds* [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis)# lsp-check-interval 240`	(Optional) Configures the time between periodic checks of the entire database to validate the checksums of the LSPs in the database. • This operation is costly in terms of CPU and so should be configured to occur infrequently.
Step 5	**lsp-gen-interval** { [**initial-wait** *initial* | **secondary-wait** *secondary* | **maximum-wait** *maximum*] **...** } [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis)# lsp-gen-interval maximum-wait 15 initial-wait 5`	(Optional) Reduces the rate of LSP generation during periods of instability in the network. Helps reduce the CPU load on the router and number of LSP transmissions to its IS-IS neighbors. • During prolonged periods of network instability, repeated recalculation of LSPs can cause an increased CPU load on the local router. Further, the flooding of these recalculated LSPs to the other Intermediate Systems in the network causes increased traffic and can result in other routers having to spend more time running route calculations.
Step 6	**lsp-mtu** *bytes* [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis)# lsp-mtu 1300`	(Optional) Sets the maximum transmission unit (MTU) size of LSPs.
Step 7	**max-lsp-lifetime** *seconds* [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis)# max-lsp-lifetime 11000`	(Optional) Sets the initial lifetime given to an LSP originated by the router. • This is the amount of time that the LSP persists in the database of a neighbor unless the LSP is regenerated or refreshed.
Step 8	**ignore-lsp-errors disable** **Example:** `RP/0/RP0/CPU0:router(config-isis)# ignore-lsp-errors disable`	(Optional) Sets the router to purge LSPs received with checksum errors.

	Command or Action	Purpose
Step 9	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3`	Enters interface configuration mode.
Step 10	**lsp-interval** *milliseconds* [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# lsp-interval 100`	(Optional) Configures the amount of time between each LSP sent on an interface.
Step 11	**csnp-interval** *seconds* [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# csnp-interval 30 level 1`	(Optional) Configures the interval at which periodic CSNP packets are sent on broadcast interfaces. • Sending more frequent CSNPs means that adjacent routers must work harder to receive them. • Sending less frequent CSNP means that differences in the adjacent routers may persist longer.
Step 12	**retransmit-interval** *seconds* [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# retransmit-interval 60`	(Optional) Configures the amount of time that the sending router waits for an acknowledgment before it considers that the LSP was not received and subsequently resends. `RP/0/RP0/CPU0:router(config-isis-if)# retransmit-interval 60`
Step 13	**retransmit-throttle-interval** *milliseconds* [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# retransmit-throttle-interval 1000`	(Optional) Configures the amount of time between retransmissions on each LSP on a point-to-point interface. • This time is usually greater than or equal to the **lsp-interval** command time because the reason for lost LSPs may be that a neighboring router is busy. A longer interval gives the neighbor more time to receive transmissions.
Step 14	**mesh-group** { *number* | **blocked** } **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# mesh-group blocked`	(Optional) Optimizes LSP flooding in NBMA networks with highly meshed, point-to-point topologies. • This command is appropriate only for an NBMA network with highly meshed, point-to-point topologies.
Step 15	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 16	**show isis interface** [*type interface-path-id* | **level** { **1** | **2** }] [**brief**] **Example:** `RP/0/RP0/CPU0:router# show isis interface GigabitEthernet 0/1/0/1 brief`	(Optional) Displays information about the IS-IS interface.
Step 17	**show isis** [**instance** *instance-id*] **database** [**level** { **1** | **2** }] [**detail** | **summary** | **verbose**] [***** | *lsp-id*] **Example:** `RP/0/RP0/CPU0:router# show isis database level 1`	(Optional) Displays the IS-IS LSP database.
Step 18	**show isis** [**instance** *instance-id*] **lsp-log** [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router# show isis lsp-log`	(Optional) Displays LSP log information.
Step 19	**show isis database-log** [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router# show isis database-log level 1`	(Optional) Display IS-IS database log information.

Configuring Nonstop Forwarding for IS-IS

This task explains how to configure your router with NSF that allows the Cisco IOS XR software to resynchronize the IS-IS link-state database with its IS-IS neighbors after a process restart. The process restart could be due to an:

- RP failover (for a warm restart)
- Simple process restart (due to an IS-IS reload or other administrative request to restart the process)
- IS-IS software upgrade

In all cases, NSF mitigates link flaps and loss of user sessions. This task is optional.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*

3. **nsf** { **cisco** | **ietf** }
4. **nsf interface-expires** *number*
5. **nsf interface-timer** *seconds*
6. **nsf lifetime** *seconds*
7. Use the **commit** or **end** command.
8. **show running-config** [*command*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **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. • You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.
Step 3	**nsf** { **cisco**	**ietf** } **Example:** `RP/0/RP0/CPU0:router(config-isis)# nsf ietf`	Enables NSF on the next restart. • Enter the **cisco** keyword to run IS-IS in heterogeneous networks that might not have adjacent NSF-aware networking devices. • Enter the **ietf** keyword to enable IS-IS in homogeneous networks where *all* adjacent networking devices support IETF draft-based restartability.
Step 4	**nsf interface-expires** *number* **Example:** `RP/0/RP0/CPU0:router(config-isis)# nsf interface-expires 1`	Configures the number of resends of an acknowledged NSF-restart acknowledgment. • If the resend limit is reached during the NSF restart, the restart falls back to a cold restart.
Step 5	**nsf interface-timer** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-isis) nsf interface-timer 15`	Configures the number of seconds to wait for each restart acknowledgment.
Step 6	**nsf lifetime** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-isis)# nsf lifetime 20`	Configures the maximum route lifetime following an NSF restart. • This command should be configured to the length of time required to perform a full NSF restart because it is the amount of time that the Routing Information Base (RIB) retains the routes during the restart.

	Command or Action	Purpose
		• Setting this value too high results in stale routes. • Setting this value too low could result in routes purged too soon.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 8	**show running-config** [*command*] **Example:** `RP/0/RP0/CPU0:router# show running-config router isis isp`	(Optional) Displays the entire contents of the currently running configuration file or a subset of that file. • Verify that "nsf" appears in the IS-IS configuration of the NSF-aware device. • This example shows the contents of the configuration file for the "isp" instance only.

Configuring ISIS-NSR

Step 1 **configure**

Example:

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

Enters XR Config mode.

Step 2 **router isis** *instance-id*

Example:

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

Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.

Step 3 **nsr**

Example:

```
RP/0/RP0/CPU0:router(config-isis)# nsr
```

Configures the NSR feature.

Step 4 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

Step 5 **show isis nsr adjacency**

Example:

```
RP/0/RP0/CPU0:router
# show isis nsr adjacency
System Id Interface SNPA State Hold Changed NSF IPv4 BFD IPv6 BFD
  R1-v1S   Nii0     *PtoP* Up   83  00:00:33 Yes  None    None
```

Displays adjacency information.

Step 6 **show isis nsr status**

Example:

```
RP/0/RP0/CPU0:router
router#show isis nsr status
IS-IS test NSR(v1a) STATUS (HA Ready):
                            V1 Standby V2 Active V2 Standby
SYNC STATUS:                   TRUE      FALSE(0) FALSE(0)
PEER CHG COUNT:                 1           0        0
UP TIME:                      00:03:12     not up    not up
```

Displays the NSR status information.

Step 7 **show isis nsr statistics**

Example:

```
RP/0/RP0/CPU0:router
router#show isis nsr statistics
IS-IS test NSR(v1a) MANDATORY STATS :
                                 V1 Active                V1 Standby         V2 Active           V2
 Standby
L1 ADJ:                               0                     0                  0
   0
L2 ADJ:                               2                     2                  0
   0
LIVE INTERFACE:                       4                     4                  0
   0
PTP INTERFACE:                        1                     1                  0
   0
LAN INTERFACE:                        2                     2                  0
   0
LOOPBACK INTERFACE:                   1                     1                  0
   0
TE Tunnel:                            1                     1                  0
   0
TE LINK:                              2                     2                  0
   0
```

```
NSR OPTIONAL STATS :
L1 LSP:                          0                   0                   0
   0
L2 LSP:                          4                   4                   0
   0
IPV4 ROUTES:                     3                   3                   0
   0
IPV6 ROUTES:                     4                   4                   0
   0
```

Shows number of ISIS adjacencies, lsps, routes, tunnels, Te links on active and standby routers.

Configuring Authentication for IS-IS

This task explains how to configure authentication for IS-IS. This task is optional.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **lsp-password** { **hmac-md5** | **text** } { **clear** | **encrypted** } *password* [**level** { **1** | **2** }] [**send-only**] [**snp send-only**]]
4. **interface** *type interface-path-id*
5. **hello-password** { **hmac-md5** | **text** } { **clear** | **encrypted** } *password* [**level** { **1** | **2** }] [**send-only**]
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **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. • You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.
Step 3	**lsp-password** { **hmac-md5** | **text** } { **clear** | **encrypted** } *password* [**level** { **1** | **2** }] [**send-only**] [**snp send-only**]] **Example:** `RP/0/RP0/CPU0:router(config-isis)# lsp-password hmac-md5 clear password1 level 1`	Configures the LSP authentication password. • The **hmac-md5** keyword specifies that the password is used in HMAC-MD5 authentication. • The **text** keyword specifies that the password uses cleartext password authentication.

	Command or Action	Purpose
		• The **clear** keyword specifies that the password is unencrypted when entered. • The **encrypted** keyword specifies that the password is encrypted using a two-way algorithm when entered. • The **level 1** keyword sets a password for authentication in the area (in Level 1 LSPs and Level SNPs). • The **level 2** keywords set a password for authentication in the backbone (the Level 2 area). • The **send-only** keyword adds authentication to LSP and sequence number protocol data units (SNPs) when they are sent. It does not authenticate received LSPs or SNPs. • The **snp send-only** keyword adds authentication to SNPs when they are sent. It does not authenticate received SNPs. **Note** To disable SNP password checking, the **snp send-only** keywords must be specified in the **lsp-password** command.
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3`	Enters interface configuration mode.
Step 5	**hello-password** { **hmac-md5** | **text** } { **clear** | **encrypted** } *password* [**level** { **1** | **2** }] [**send-only**] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)#hello-password text clear mypassword`	Configures the authentication password for an IS-IS interface.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Keychains for IS-IS

This task explains how to configure keychains for IS-IS. This task is optional.

Keychains can be configured at the router level (**lsp-password** command) and at the interface level (**hello-password** command) within IS-IS. These commands reference the global keychain configuration and instruct the IS-IS protocol to obtain security parameters from the global set of configured keychains. The router-level configuration (**lsp-password** command) sets the keychain to be used for all IS-IS LSPs generated by this router, as well as for all Sequence Number Protocol Data Units (SN PDUs). The keychain used for HELLO PDUs is set at the interface level, and may be set differently for each interface configured for IS-IS.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **l sp-password keychain** *keychain-name* [**level** { **1** | **2** }] [**send-only**] [**snp send-only**]
4. **interface** *type interface-path-id*
5. **h ello-password keychain** *keychain-name* [**level** { **1** | **2** }] [**send-only**]
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **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. • You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.
Step 3	**l sp-password keychain** *keychain-name* [**level** { **1** | **2** }] [**send-only**] [**snp send-only**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# lsp-password keychain isis_a level 1`	Configures the keychain.
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3`	Enters interface configuration mode.

	Command or Action	Purpose
Step 5	**h ello-password keychain** *keychain-name* [**level** { **1** | **2** }] [**send-only**] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)#hello-password keychain isis_b`	Configures the authentication password for an IS-IS interface.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring MPLS Traffic Engineering for IS-IS

This task explains how to configure IS-IS 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 IS-IS can use, and troubleshoot MPLS TE, see *Implementing MPLS Traffic Engineering on MPLS Configuration Guide for Cisco NCS 6000 Series Routers*

Before you begin

Your network must support the MPLS Cisco IOS XR software feature before you enable MPLS TE for IS-IS on your router.

Note You must enter the commands in the following task list on every IS-IS router in the traffic-engineered portion of your network.

Note MPLS traffic engineering currently does not support routing and signaling of LSPs over unnumbered IP links. Therefore, do not configure the feature over those links.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **address-family** { **ipv4** | **ipv6** } [**unicast**]

4. **mpls traffic-eng level** { **1** | **2** }
5. **mpls traffic-eng router-id** { *ip-address* | *interface-name interface-instance* }
6. **metric-style wide** [**level** { **1** | **2** }]
7. Use the **commit** or **end** command.
8. **show isis** [**instance** *instance-id*] **mpls traffic-eng tunnel**
9. **show isis** [**instance** *instance-id*] **mpls traffic-eng adjacency-log**
10. **show isis** [**instance** *instance-id*] **mpls traffic-eng advertisements**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **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. • You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.
Step 3	**address-family** { **ipv4** | **ipv6** } [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis)#address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.
Step 4	**mpls traffic-eng level** { **1** | **2** } **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# mpls traffic-eng level 1`	Configures a router running IS-IS to flood MPLS TE link information into the indicated IS-IS level.
Step 5	**mpls traffic-eng router-id** { *ip-address* | *interface-name interface-instance* } **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# mpls traffic-eng router-id loopback0`	Specifies that the MPLS TE router identifier for the node is the given IP address or an IP address associated with the given interface.
Step 6	**metric-style wide** [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide level 1`	Configures a router to generate and accept only wide link metrics in the Level 1 area.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 8	**show isis** [**instance** *instance-id*] **mpls traffic-eng tunnel** **Example:** `RP/0/RP0/CPU0:router# show isis instance isp mpls traffic-eng tunnel`	(Optional) Displays MPLS TE tunnel information.
Step 9	**show isis** [**instance** *instance-id*] **mpls traffic-eng adjacency-log** **Example:** `RP/0/RP0/CPU0:router# show isis instance isp mpls traffic-eng adjacency-log`	(Optional) Displays a log of MPLS TE IS-IS adjacency changes.
Step 10	**show isis** [**instance** *instance-id*] **mpls traffic-eng advertisements** **Example:** `RP/0/RP0/CPU0:router# show isis instance isp mpls traffic-eng advertisements`	(Optional) Displays the latest flooded record from MPLS TE.

Tuning Adjacencies for IS-IS

This task explains how to enable logging of adjacency state changes, alter the timers for IS-IS adjacency packets, and display various aspects of adjacency state. Tuning your IS-IS adjacencies increases network stability when links are congested. This task is optional.

For point-to-point links, IS-IS sends only a single hello for Level 1 and Level 2, which means that the level modifiers are meaningless on point-to-point links. To modify hello parameters for a point-to-point interface, omit the specification of the level options.

The options configurable in the interface submode apply only to that interface. By default, the values are applied to both Level 1 and Level 2.

The **hello-password** command can be used to prevent adjacency formation with unauthorized or undesired routers. This ability is particularly useful on a LAN, where connections to routers with which you have no desire to establish adjacencies are commonly found.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **log adjacency changes**
4. **interface** *type interface-path-id*
5. **hello-padding** { **disable** | **sometimes** } [**level** { **1** | **2** }]
6. **hello-interval** *seconds* [**level** { **1** | **2** }]
7. **hello-multiplier** *multiplier* [**level** { **1** | **2** }]
8. **h ello-password** { **hmac-md5** | **text** } { **clear** | **encrypted** } *password* [**level** { **1** | **2** }] [**send-only**]
9. Use the **commit** or **end** command.
10. **show isis** [**instance** *instance-id*] **adjacency** *t ype interface- path-id*] [**detail**] [**systemid** *system-id*]
11. show isis adjacency-log
12. **show isis** [**instance** *instance-id*] **interface** [*type interface-path-id*] [**brief** | **detail**] [**level** { **1** | **2** }]
13. **show isis** [**instance** *instance-id*] **neighbors** [*interface-type interface-instance*] [**summary**] [**detail**] [**systemid** *system-id*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **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. • You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.
Step 3	**log adjacency changes** **Example:** `RP/0/RP0/CPU0:router(config-isis)# log adjacency changes`	Generates a log message when an IS-IS adjacency changes state (up or down).
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3`	Enters interface configuration mode.
Step 5	**hello-padding** { **disable** | **sometimes** } [**level** { **1** | **2** }]	Configures padding on IS-IS hello PDUs for an IS-IS interface on the router.

	Command or Action	Purpose
	Example: `RP/0/RP0/CPU0:router(config-isis-if)# hello-padding sometimes`	• Hello padding applies to only this interface and not to all interfaces.
Step 6	**hello-interval** *seconds* [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)#hello-interval 6`	Specifies the length of time between hello packets that the software sends.
Step 7	**hello-multiplier** *multiplier* [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# hello-multiplier 10`	Specifies the number of IS-IS hello packets a neighbor must miss before the router should declare the adjacency as down. • A higher value increases the networks tolerance for dropped packets, but also may increase the amount of time required to detect the failure of an adjacent router. • Conversely, not detecting the failure of an adjacent router can result in greater packet loss.
Step 8	**h ello-password** { **hmac-md5** | **text** } { **clear** | **encrypted** } *password* [**level** { **1** | **2** }] [**send-only**] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# hello-password text clear mypassword`	Specifies that this system include authentication in the hello packets and requires successful authentication of the hello packet from the neighbor to establish an adjacency.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 10	**show isis** [**instance** *instance-id*] **adjacency** *t ype interface- path-id*] [**detail**] [**systemid** *system-id*] **Example:** `RP/0/RP0/CPU0:router# show isis instance isp adjacency`	(Optional) Displays IS-IS adjacencies.

	Command or Action	Purpose
Step 11	show isis adjacency-log **Example:** `RP/0/RP0/CPU0:router# show isis adjacency-log`	(Optional) Displays a log of the most recent adjacency state transitions.
Step 12	**show isis** [**instance** *instance-id*] **interface** [*type interface-path-id*] [**brief** | **detail**] [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router# show isis interface GigabitEthernet 0/1/0/1 brief`	(Optional) Displays information about the IS-IS interface.
Step 13	**show isis** [**instance** *instance-id*] **neighbors** [*interface-type interface-instance*] [**summary**] [**detail**] [**systemid** *system-id*] **Example:** `RP/0/RP0/CPU0:router# show isis neighbors summary`	(Optional) Displays information about IS-IS neighbors.

Setting SPF Interval for a Single-Topology IPv4 and IPv6 Configuration

This task explains how to make adjustments to the SPF calculation to tune router performance. This task is optional.

Because the SPF calculation computes routes for a particular topology, the tuning attributes are located in the router address family configuration submode. SPF calculation computes routes for Level 1 and Level 2 separately.

When IPv4 and IPv6 address families are used in a single-topology mode, only a single SPF for the IPv4 topology exists. The IPv6 topology "borrows" the IPv4 topology; therefore, no SPF calculation is required for IPv6. To tune the SPF calculation parameters for single-topology mode, configure the **address-family ipv4 unicast** command.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **address-family** { **ipv4** | **ipv6** } [**unicast**]
4. **spf-interval** {[**initial-wait** *initial* | **secondary-wait** *secondary* | **maximum-wait** *maximum*] ...} [**level** { **1** | **2** }]
5. Use the **commit** or **end** command.
6. **show isis** [**instance** *instance-id*] [[**ipv4** | **ipv6** | **afi-all**] [**unicast** | **safi-all**]] **spf-log** [**level** { **1** | **2** }] [**fspf** | **prc** | **nhc**] [**detail** | **verbose**] [**last** *number* | **first** *number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **Example:** `Router(config)# router isis isp`	Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode. • You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.
Step 3	**address-family** { **ipv4** | **ipv6** } [**unicast**] **Example:** `Router(config-isis)#address-family ipv4 unicast`	Specifies the IPv4or IPv6 address family, and enters router address family configuration mode.
Step 4	**spf-interval** {[**initial-wait** *initial* | **secondary-wait** *secondary* | **maximum-wait** *maximum*] ...} [**level** { **1** | **2** }] **Example:** `Router(config-isis-af)# spf-interval initial-wait 10 maximum-wait 30`	(Optional) Controls the minimum time between successive SPF calculations. • This value imposes a delay in the SPF computation after an event trigger and enforces a minimum elapsed time between SPF runs. • If this value is configured too low, the router can lose too many CPU resources when the network is unstable. • Configuring the value too high delays changes in the network topology that result in lost packets.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 6	**show isis** [**instance** *instance-id*] [[**ipv4** | **ipv6** | **afi-all**] [**unicast** | **safi-all**]] **spf-log** [**level** { **1** | **2** }] [**fspf** | **prc** | **nhc**] [**detail** | **verbose**] [**last** *number* | **first** *number*] **Example:**	(Optional) Displays how often and why the router has run a full SPF calculation.

	Command or Action	Purpose
	`Router# show isis instance 1 spf-log ipv4`	

Customizing Routes for IS-IS

This task explains how to perform route functions that include injecting default routes into your IS-IS routing domain and redistributing routes learned in another IS-IS instance. This task is optional.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **set-overload-bit** [**on-startup** { *delay* | **wait-for-bgp** }] [**level** { **1** | **2** }]
4. **address-family** { **ipv4** | **ipv6** } [**unicast**]
5. **default-information originate** [**route-policy** *route-policy-name*]
6. **redistribute isis** *instance* [**level-1** | **level-2** | **level-1-2**] [**metric** *metric*] [**metric-type** { **internal** | **external** }] [**policy** *policy-name*]
7. Do one of the following:
   - **summary-prefix** *address* / *prefix-length* [**level** { **1** | **2** }]
   - **summary-prefix** *ipv6-prefix* / *prefix-length* [**level** { **1** | **2** }]
8. **maximum-paths** *route-number*
9. **distance** *weight* [*address* / *prefix-length* [*route-list-name*]]
10. **attached-bit send { always-set | never-set }**
11. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis isp`	Enables IS-IS routing for the specified routing process, and places the router in router configuration mode. • By default, all IS-IS instances are automatically Level 1 and Level 2. You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.
Step 3	**set-overload-bit** [**on-startup** { *delay* | **wait-for-bgp** }] [**level** { **1** | **2** }]	(Optional) Sets the overload bit.

	Command or Action	Purpose
	Example: `RP/0/RP0/CPU0:router(config-isis)# set-overload-bit`	**Note** The configured overload bit behavior does not apply to NSF restarts because the NSF restart does not set the overload bit during restart. **Note** When the **wait-for-bgp** keyword is used, BGP will not send convergence notification, if it does not have the IPv4 or IPv6 address-family configured in the default VRF.
Step 4	**address-family** { **ipv4** | **ipv6** } [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.
Step 5	**default-information originate** [**route-policy** *route-policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# default-information originate`	(Optional) Injects a default IPv4 or IPv6 route into an IS-IS routing domain. • The **route-policy** keyword and *route-policy-name* argument specify the conditions under which the IPv4 or IPv6 default route is advertised. • If the **route-policy** keyword is omitted, then the IPv4 or IPv6 default route is unconditionally advertised at Level 2.
Step 6	**redistribute isis** *instance* [**level-1** | **level-2** | **level-1-2**] [**metric** *metric*] [**metric-type** { **internal** | **external** }] [**policy** *policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# redistribute isis 2 level-1`	(Optional) Redistributes routes from one IS-IS instance into another instance. • In this example, an IS-IS instance redistributes Level 1 routes from another IS-IS instance.

Step 7	Do one of the following:
	• **summary-prefix** *address* / *prefix-length* [**level** { **1** | **2** }]
	• **summary-prefix** *ipv6-prefix* / *prefix-length* [**level** { **1** | **2** }]
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# summary-prefix 10.1.0.0/16 level 1`
	or
	`RP/0/RP0/CPU0:router(config-isis-af)# summary-prefix 3003:xxxx::/24 level 1`
		(Optional) Allows a Level 1-2 router to summarize Level 1 IPv4 and IPv6 prefixes at Level 2, instead of advertising the Level 1 prefixes directly when the router advertises the summary.
		• This example specifies an IPv4 address and mask.
		or
		• This example specifies an IPv6 prefix, and the command must be in the form documented in RFC 2373 in which the address is specified in hexadecimal using 16-bit values between colons.

	Command or Action	Purpose
		• Note that IPv6 prefixes must be configured only in the IPv6 router address family configuration submode, and IPv4 prefixes in the IPv4 router address family configuration submode.
Step 8	**maximum-paths** *route-number* **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# maximum-paths 16`	(Optional) Configures the maximum number of parallel paths allowed in a routing table.
Step 9	**distance** *weight* [*address / prefix-length* [*route-list-name*]] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# distance 90`	(Optional) Defines the administrative distance assigned to routes discovered by the IS-IS protocol. • A different administrative distance may be applied for IPv4 and IPv6.
Step 10	**attached-bit send { always-set | never-set }** **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# attached-bit send always-set`	(Optional) Configures an IS-IS instance with an attached bit in the Level 1 LSP.
Step 11	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring MPLS LDP IS-IS Synchronization

This task explains how to enable Multiprotocol Label Switching (MPLS) Label Distribution Protocol (LDP) IS-IS synchronization. MPLS LDP synchronization can be enabled for an address family under interface configuration mode. Only IPv4 unicast address family is supported. This task is optional.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **interface** *type interface-path-id*
4. **address-family ipv4 unicast**

5. **mpls ldp sync** [**level** { **1** | **2** }]
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis isp`	Enables IS-IS routing for the specified routing process, and places the router in router configuration mode. • By default, all IS-IS instances are automatically Level 1 and Level 2. You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.
Step 3	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3`	Enters interface configuration mode.
Step 4	**address-family ipv4 unicast** **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters router address family configuration mode.
Step 5	**mpls ldp sync** [**level** { **1**	**2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-if-af)# mpls ldp sync level 1`	Enables MPLS LDP synchronization for the IPv4 address family under interface GigabitEthernet 0/1/0/3.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Tagging IS-IS Interface Routes

This optional task describes how to associate a tag with a connected route of an IS-IS interface.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **address-family** { **ipv4** | **ipv6** } [**unicast**]
4. **metric-style wide** [**transition**] [**level** { **1** | **2** }]
5. **exit**
6. **interface** *type number*
7. **address-family** { **ipv4** | **ipv6** } [**unicast**]
8. **tag** *tag*
9. Use the **commit** or **end** command.
10. **show isis** [**ipv4** | **ipv6** | **afi-all**] [**unicast** | **safi-all**] **route** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis isp`	Enables IS-IS routing for the specified routing process, and places the router in router configuration mode. In this example, the IS-IS instance is called isp.
Step 3	**address-family** { **ipv4**	**ipv6** } [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.
Step 4	**metric-style wide** [**transition**] [**level** { **1**	**2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide level 1`	Configures a router to generate and accept only wide link metrics in the Level 1 area.
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# exit`	Exits router address family configuration mode, and returns the router to router configuration mode.

	Command or Action	Purpose
Step 6	**interface** *type number* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3`	Enters interface configuration mode.
Step 7	**address-family** { **ipv4** | **ipv6** } [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters address family configuration mode.
Step 8	**tag** *tag* **Example:** `RP/0/RP0/CPU0:router(config-isis-if-af)# tag 3`	Sets the value of the tag to associate with the advertised connected route.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 10	**show isis** [**ipv4** | **ipv6** | **afi-all**] [**unicast** | **safi-all**] **route** [**detail**] **Example:** `RP/0/RP0/CPU0:router(config-isis-if-af)# show isis ipv4 route detail`	Displays tag information. Verify that all tags are present in the RIB.

Setting the Priority for Adding Prefixes to the RIB

This optional task describes how to set the priority (order) for which specified prefixes are added to the RIB. The prefixes can be chosen using an access list (ACL), prefix list, or by matching a tag value.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **address-family** { **ipv4** \| **ipv6** } [**unicast**]
4. **metric-style wide** [**transition**] [**level** { **1** \| **2** }]

5. **spf prefix-priority** [**level** { **1** | **2** }] { **critical** | **high** | **medium** } { *access-list-name* | **tag** *tag* }
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis isp`	Enables IS-IS routing for the specified routing process, and places the router in router configuration mode. In this example, the IS-IS instance is called isp.
Step 3	**address-family** { **ipv4** | **ipv6** } [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.
Step 4	**metric-style wide** [**transition**] [**level** { **1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide level 1`	Configures a router to generate and accept only wide-link metrics in the Level 1 area.
Step 5	**spf prefix-priority** [**level** { **1** | **2** }] { **critical** | **high** | **medium** } { *access-list-name* | **tag** *tag* } **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# spf prefix-priority high tag 3`	Installs all routes tagged with the value 3 first.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring IP Fast Reroute Loop-free Alternate

This optional task describes how to enable the IP fast reroute (IPFRR) loop-free alternate (LFA) computation to converge traffic flows around link failures.

Note To enable node protection on broadcast links, IPFRR and bidirectional forwarding detection (BFD) must be enabled on the interface under IS-IS.

Before you begin

Note IPFRR is supported on the Cisco IOS XR. IPv4 address families and single-level interfaces are supported.

Multiprotocol Label Switching (MPLS) FRR and IPFRR cannot be configured on the same interface simultaneously.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **interface** *type interface-path-id*
4. **circuit-type** { **level-1** | **level-1-2** | **level-2-only** }
5. **address-family ipv4 unicast**
6. **ipfrr lfa** { **level** { **1** | **2** }}
7. **ipfrr lfa exclude interface** *type interface-path-id*
8. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis isp`	Enables IS-IS routing for the specified routing process, and places the router in router configuration mode. In this example, the IS-IS instance is called isp.
Step 3	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3`	Enters interface configuration mode.

	Command or Action	Purpose
Step 4	**circuit-type** { **level-1** | **level-1-2** | **level-2-only** } **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# circuit-type level-1`	(Optional) Configures the type of adjacency.
Step 5	**address-family ipv4 unicast** **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast`	Specifies the IPv4 address family, and enters router address family configuration mode.
Step 6	**ipfrr lfa** { **level** { **1** | **2** }} **Example:** `RP/0/RP0/CPU0:router(config-isis-if-af)# ipfrr lfa level 1`	Specifies the IP fast reroute loop-free alternate computation on link or node failures.
Step 7	**ipfrr lfa exclude interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-isis-if-af)#ipfrr lfa exclude interface POS 0/1/0/4`	(Optional) Excludes an interface from the IP fast reroute loop-free alternate computation.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure Remote Loop-Free Alternate Paths for IS-IS

IP Fast Re-Route (FRR) with Loop-Free Alternate (LFA) paths uses only the immediate next hops (directly-connected neighbors) as backups to a destination. Though this works well in most topologies, it fails in ring topologies, because the immediate next hop is likely to use the source router itself to forward traffic, thereby creating loops.

As a result, during link failures, extended LFA or Remote LFA (RLFA) is required to enable the router to use a non-directly connected next hop as backup. RLFA ensures that this nearest non-connected neighbor does not loop back to the source router.

RLFA uses an LDP tunnel between source and backup routers and is implemented in IPv4, IPv6, and MPLS networks. When the network converges, micro-loops can form due to inconsistencies in the forwarding tables of different routers. RLFA also provides a mechanism to avoid micro-loops.

Remote LFA is suitable for incremental deployment within a network, including a network that is already deploying LFA.

Remote LFA Process

The remote LFA process on the source router involves:

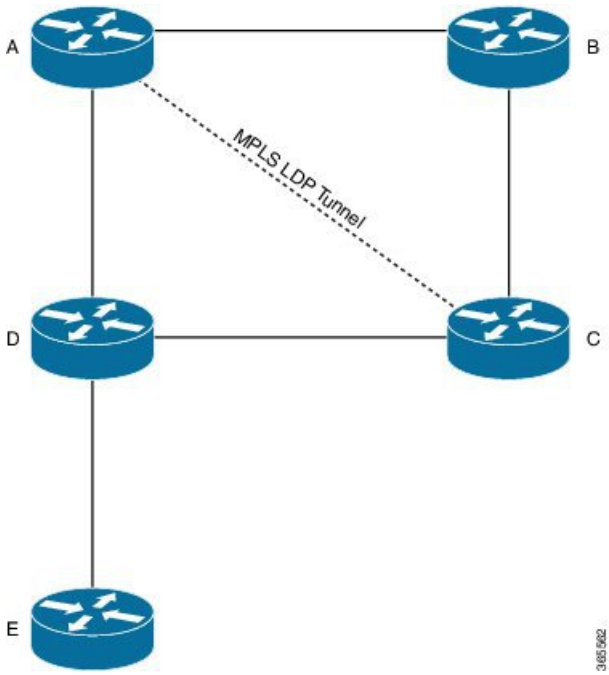
1. For a protected link between the source router and the immediate next hop, the IGP (OSPF or IS-IS) computes the remote LFA backup path by using the PQ algorithm.
2. The IGP updates the RIB table with the remote LFA path information.
3. LDP establishes a session with the remote router to exchange labels for prefixes.
4. LDP sets up MPLS forwarding for the protected prefix, and the corresponding backup path.
5. On link failure, and fast reroute trigger, the remote LFA backup path is activated with less than 50 millisecond convergence time.
6. The backup path is active until the IGP converges to the new primary path.

Remote LFA Topology

Consider the topology in the following figure. The best path from Source Router A to Destination Router E is through Router D. The best backup path for this route would be: Router A -> B -> C -> D -> Router E. Because Router B uses Router A as its primary next hop, this backup path cannot be used and LFA fails in this topology.

As a result, by implementing RLFA, Router A calculates its nearest non-directly connected neighbor that does not route back to it. In this example, Router A chooses Router C as its RLFA backup, and uses LDP to establish a tunnel between them. Traffic from Router A is routed to Router C on fast reroute trigger.

Figure 14: Remote LFA Topology



Configuration

To configure RLFA with IS-IS, use the following steps.

1. Configure the interface(s) of the router.

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#interface gigabitEthernet 0/0/0/1
RP/0/RP0/CPU0:router(config-if)#ipv4 address 10.1.1.1 255.255.255.0
RP/0/RP0/CPU0:router(config-if)#no shutdown
RP/0/RP0/CPU0:router(config-if)#exit
RP/0/RP0/CPU0:router(config)#interface gigabitEthernet 0/0/0/2
RP/0/RP0/CPU0:router(config-if)#ipv4 address 10.10.10.1 255.255.255.0
RP/0/RP0/CPU0:router(config-if)#no shutdown
RP/0/RP0/CPU0:router(config-if)#exit
RP/0/RP0/CPU0:router(config)#interface loopback0
RP/0/RP0/CPU0:router(config-if)#ipv4 address 10.10.10.1 255.255.255.255
RP/0/RP0/CPU0:router(config-if)#no shutdown
RP/0/RP0/CPU0:router(config-if)#exit
```

2. Configure IS-IS.

```
RP/0/RP0/CPU0:router(config)#router isis rlfa
RP/0/RP0/CPU0:router(config-isis)#net 10.0001.0001.0001.00
cted
```

3. Configure RLFA for IS-IS.

```
RP/0/RP0/CPU0:router(config-isis)# interface gigabitEthernet 0/0/0/1
RP/0/RP0/CPU0:router(config-isis-if)#address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)#fast-reroute per-prefix remote-lfa tunnel mpls-ldp
RP/0/RP0/CPU0:router(config-isis-if-af)#fast-reroute per-prefix remote-lfa maximum-metric
 20
RP/0/RP0/CPU0:router(config-isis-if-af)#exit
```

```
RP/0/RP0/CPU0:router(config-isis-if)# exit
RP/0/RP0/CPU0:router(config-isis)# microloop avoidance protected
```

The **maximum-metric** command is required to limit the range of remote LFAs. The **microloop avoidance** command is used to avoid micro-loops by delaying the convergence of all or protected prefixes (as configured in this example). For more information on these commands, see the *Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference*.

4. Commit your configuration.

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

5. Confirm your configuration.

```
RP/0/RP0/CPU0:router(config)#show running-config
...
!
interface GigabitEthernet0/0/0/1
 ipv4 address 10.1.1.1 255.255.255.0
!
interface GigabitEthernet0/0/0/2
 ipv4 address 10.10.10.1 255.255.255.0
 shutdown

interface loopback0
 ipv4 address 10.1.1.1 255.255.255.255
 shutdown
...
!
router isis rlfa
 microloop avoidance protected
 net 10.0001.0001.0001.00
  interface GigabitEthernet0/0/0/1
  address-family ipv4 unicast
   fast-reroute per-prefix remote-lfa tunnel mpls-ldp
   fast-reroute per-prefix remote-lfa maximum-metric 20
  !
 !
!
...
```

Sample Verification Outputs

You can run the show commands mentioned in this section to verify whether RLFA is operational in your network. This section lists the sample outputs that are retrieved depending on your network topology.

RIB Verification

Verify the presence of the remote backup paths in the RIB.

```
RP/0/RP0/CPU0:router#show route 10.1.1.1/32

Routing entry for 10.1.1.1/32
  Known via "isis rlfa", distance 110, metric 20, type extern 2
  Installed Sep  8 15:18:33.214 for 2d00h
  Routing Descriptor Blocks
    13.0.0.3, from 10.1.1.1, via GigabitEthernet0/0/0/1, Protected
    ...
    131.1.1.4, from 10.10.10.1 via GigabitEthernet0/0/0/2, Backup (remote)
    ...
```

```
RP/0/RP0/CPU0:router#show route 10.1.1.1/32 detail

Routing entry for 10.1.1.1/32
  Known via "isis rlfa", distance 110, metric 20, type extern 2
  Installed Sep  8 15:18:33.214 for 2d00h
  Routing Descriptor Blocks
    13.0.0.3, from 10.1.1.1, via GigabitEthernet0/0/0/1, Protected
      Path id: 1     Path ref count: 0
      Backup path id: 33
    ...
    131.1.1.4, from 10.10.10.1 via GigabitEthernet0/0/0/2, Backup (remote)
     Remote LFA is 4.4.4.4
      Path id: 33   Path refcount:1
   ...
```

IS-IS Verification

Verify the presence of the remote backup paths in the IS-IS topology table.

```
RP/0/RP0/CPU0:router#show isis fast-reroute
...
IS-IS 1 IPv4 Unicast FRR backups

Codes: L1 - level 1, L2 - level 2, ia - interarea (leaked into level 1)
       df - level 1 default (closest attached router), su - summary null
       C - connected, S - static, R - RIP, B - BGP, O - OSPF
       E - EIGRP, A - access/subscriber
       i - IS-IS (redistributed from another instance)
       D - Downstream, LC - Line card disjoint, NP - Node protecting
       P - Primary path, SRLG - SRLG disjoint, TM - Total metric via backup

Maximum parallel path count: 8

L1 3.3.3.3/32 [20/115]
     via 30.1.1.2, GigabitEthernet0/0/0/2, rt3
      FRR backup via 20.1.1.2, GigabitEthernet0/0/0/1, rt2   /*□ Local LFA backup display*/

L1 40.1.1.0/24 [20/115]
     via 30.1.1.2, GigabitEthernet0/0/0/2, rt3
       Remote FRR backup via rt5 [12.12.12.12], via 20.1.1.2, GigabitEthernet0/0/0/1, rt2

   /* Remote LFA backup display */
```

FIB Verification

Verify the presence of remote backup paths in the FIB.

```
RP/0/RP0/CPU0:router# show cef 10.1.1.1 detail

10.1.1.1/32, version 6, internal 0x4004001 (ptr 0x1d5977f4) [1], 0x0 (0x1d563344), 0x450
(0x1dbab050)
 Updated Apr 27 13:36:03.013
 remote adjacency to GigabitEthernet0/2/0/0
 Prefix Len 32, traffic index 0, precedence routine (0), priority 3
  gateway array (0x1d42b878) reference count 3, flags 0x28000d0, source lsd (3), 0 backups
                [2 type 5 flags 0x10101 (0x1da3c230) ext 0x0 (0x0)]
  LW-LDI[type=5, refc=3, ptr=0x1d563344, sh-ldi=0x1da3c230]
   via 2.2.2.3, GigabitEthernet0/2/0/0, 0 dependencies, weight 0, class 0, protected [flags
 0x400]
    path-idx 0 bkup-idx 1 [0x1dc560cc 0x0]
    next hop 2.2.2.3
     local label 16011      labels imposed {16001}
   via 1.1.1.2, GigabitEthernet0/1/0/0, 2 dependencies, weight 0, class 0, backup (remote)
```

```
 [flags 0x300]
    path-idx 1 [0x1dc22110 0x0]
    next hop 1.1.1.2
    remote adjacency
     local label 16011      labels imposed {16002 16003}

    Load distribution: 0 (refcount 2)

    Hash  OK  Interface                 Address
    0     Y   GigabitEthernet0/2/0/0    remote
```

```
RP/0/RP0/CPU0:router# show mpls forwarding prefix 10.1.1.1/32

Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
16011  16001       10.1.1.1/32         SI0/2/CPU0   2.2.2.3         0
       {16002      10.1.1.1/32         FI0/1/CPU0   1.1.1.2         0
       16003 }
```

```
RP/0/RP0/CPU0:router# show mpls forwarding prefix 10.1.1.1/32 detail

Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
16011  16001       10.1.1.1/32         SI0/2/CPU0   2.2.2.3         0
     Updated Apr 29 14:25:09.770
     Path Flags: 0x400 [ BKUP-IDX:1 (0x1dc460cc) ]
     Version: 5, Priority: 3
     MAC/Encaps: 0/4, MTU: 8000
     Label Stack (Top -> Bottom): { 16001 }
     Packets Switched: 0

       { 16002         10.1.1.1/32         FI0/1/CPU0   1.1.1.2         0
         16003 }
     Updated Apr 29 14:25:09.770
     Path Flags: 0x300 [ IDX:1 BKUP-REMOTE ]
     MAC/Encaps: 0/4, MTU: 1500
     Label Stack (Top -> Bottom): { 16002  16003 }
     Packets Switched: 0
```

```
RP/0/RP0/CPU0:router# show mpls forwarding prefix 10.1.1.1/32

Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
16011  16001       10.1.1.1/32         GI0/1/CPU0   2.2.2.3         0
      {16002       10.1.1.1/32         GI0/2/CPU0   1.1.1.2         0
       16003 }
```

```
RP/0/RP0/CPU0:router# show mpls forwarding prefix 10.1.1.1/32 detail

Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
16011  16001       10.1.1.1/32         GI0/1/CPU0   2.2.2.3         0
     Updated Apr 29 14:25:09.770
     Path Flags: 0x400 [ BKUP-IDX:1 (0x1dc460cc) ]
     Version: 5, Priority: 3
```

```
     MAC/Encaps: 0/4, MTU: 8000
     Label Stack (Top -> Bottom): { 16001 }
     Packets Switched: 0

       { 16002         10.1.1.1/32         GI0/2/CPU0   1.1.1.2         0
         16003 }
     Updated Apr 29 14:25:09.770
     Path Flags: 0x300 [ IDX:1 BKUP-REMOTE ]
     MAC/Encaps: 0/4, MTU: 1500
     Label Stack (Top -> Bottom): { 16002  16003 }
     Packets Switched: 0
-------------------------------------------------------------------------------------------
RP/0/RP0/CPU0:router# show cef fast-reroute

Prefix              Next Hop            Interface
10.1.1.1/32      2.18.6.2            Bundle-Ether100
                    13.1.11.2           GigabitEthernet0/6/1/9.11
10.2.1.1/32      2.18.6.2            Bundle-Ether100
                    13.1.11.2           GigabitEthernet0/6/1/9.11
10.3.1.1/32      2.18.6.2            Bundle-Ether100
                    13.1.11.2           GigabitEthernet0/6/1/9.11
-------------------------------------------------------------------------------------------
```

MPLS LDP Verification

Verify the presence of remote backup paths in the MPLS LDP forwarding database.

```
RP/0/RP0/CPU0:router# show mpls ldp forwarding 10.1.1.1/32
Codes:
  - = GR label recovering, (!) = LFA FRR Pure Backup path
  {} = Label stack with multi-line output for a routing path
  G = GR, S = Stale, R = Remote LFA FRR Backup

Prefix           Label   Label          Outgoing     Next Hop            Flags
                 In      Out            Interface                        G S R
---------------- ------- -------------- ------------ ------------------- -----
10.1.1.1/32       16001   16002          Gi0/0/0/1   12.1.0.2
                         {16003          Gi0/0/0/2   13.1.0.3        (!) G    R
                          16004 }                   (4.4.4.4)           G

-------------------------------------------------------------------------------

RP/0/RP0/CPU0:router# show mpls ldp forwarding 10.1.1.1/32 detail

Codes:
  - = GR label recovering, (!) = LFA FRR Pure Backup path
  {} = Label stack with multi-line output for a routing path
  G = GR, S = Stale, R = Remote LFA FRR Backup

Prefix           Label   Label          Outgoing     Next Hop            Flags
                 In      Out            Interface                        G S R
---------------- ------- -------------- ------------ ------------------- -----
10.1.1.1/32       16001   16002          Gi0/0/0/1    12.1.0.2
                         [ Protected; path-id 1 backup-path-id 33;
                           peer 2.2.2.2:0 ]
                         { 16003        Gi0/0/0/2    13.1.0.3        (!) G   R
                           16004 }                    (4.4.4.4)           G
                         [ Backup (remote); path-id 33; peer 3.3.3.3:0
                           remote LFA 4.4.4.4, remote peer 4.4.4.4:0 ]
  Routing update   : Apr 23 17:22:33.102 (00:08:02 ago)
  Forwarding update: Apr 23 17:22:47.183 (00:07:48 ago)
-------------------------------------------------------------------------------
```

```
RP/0/RP0/CPU0:router# show mpls ldp forwarding summary

Forwarding Server (LSD):
  Connected: Yes
  Forwarding State Holdtime: 120 sec
Forwarding States:
…
  Rewrites:
Prefix:
      Total: 5 (4 with ECMP, 1 LFA FRR protected)
      Labelled:
…
    Paths:
      Total: 10 (1 backup [1 remote], 2 LFA FRR protected)
      Labelled: 10 (1 backup)
```

MPLS LSD Verification

Verify the presence of remote backup paths in the MPLS LSD forwarding database.

```
RP/0/RP0/CPU0:router# show mpls lsd forwarding labels 16001

In_Label, (ID), Path_Info: <Type>
16001, (IPv4, 'default':4U, 10.1.1.1/32), 2 Paths
   1/2: IPv4-rLFA, 'default':4U, Gi0/0/0/1, nh=12.1.0.2, lbl=16002, tun_id=0,
            flags=0x0 () [Protected]
   2/2: IPv4-rLFA, 'default':4U, Gi0/0/0/2, nh=13.1.0.3, lbls={ 16003, 16004 }, tun_id=0,
    flags={ 0x4008 (Retain, Remote-Backup), 0x8 (Retain) } [Backup]
------------------------------------------------------------------------------------------

RP/0/RP0/CPU0:router# show mpls lsd forwarding labels 16001 detail

In_Label, (ID), Path_Info: <Type>
16001, (IPv4, 'default':4U, 10.1.1.1/32), 2 Paths
   1/2: IPv4-rLFA, 'default':4U, Gi0/0/0/1, nh=12.1.0.2, lbl=16002, tun_id=0,
            flags=0x0 () [Protected]
            path-id=1, backup-path-id=33
   2/2: IPv4-rLFA, 'default':4U, Gi0/0/0/2, nh=13.1.0.3, lbls={ 16003, 16004 }, tun_id=0,
            flags={ 0x4008 (Retain, Remote-Backup), 0x8 (Retain) } [Backup]
            path-id=33, backup-path-id=0
BCDL priority:3, LSD queue:15, version:103
      Installed Apr 23 17:22:47.183 (00:17:09 ago)
------------------------------------------------------------------------------------------

RP/0/RP0/CPU0:router# show mpls lsd forwarding summary

Messages: 22
Forwarding updates: 34
Rewrites: 9
  FPIs:
    Label: 9
    IPv4: 5
..
  MOIs: 13
    IPv4 paths: 10 (1 backup [1 remote], 2 protected)
..
    IP subscriber: 0
```

Ingress Forwarding Chain Verification (Backup over Non-TE)

Verify the presence of remote backup paths in the ingress forwarding chain.

```
RP/0/RP0/CPU0:router#show cef 10.1.1.1 hard in det loc 0/1/CPU0
Sat May 12 05:16:47.929 UTC
10.10.10.10/32, version 5475, internal 0x4004001 (ptr 0x5f9582f0) [1], 0x0 (0x5f29cf94),
0x450 (0x6058e35c)
 Updated May 12 05:05:51.294
...

 Print Flags: 00000000

  INGRESS PLU
   SW: 08300200 00000004 00040000 00263b00
   HW: 08300200 00000004 00040000 00263b00
  entry_type:         FWD    rpf ptr:     0x00300200
  prefix len:           1    BGP policy a/c:       0
QoS group:            0    BAO id:               0
  num entries:          1    next ptr:    0x0000263b
  label ptr:   0x00000000    Label(0) Ptr(0)

    Load info:
    Flag: 0x00860001
    TLU Channel: 1 Addr: 0x0000263b
    ENTRY        0
      SW: 00000000 00712426 b4040002 00000001
      HW: 00000000 00712426 b4040002 00000001
    PBTS:              0    extra l3li:             0
    entry type:      FWD    next ptr:     0x00712426
    is label:    0    is label ptr:    0
    num of entries:    1
    tunnel_encap_ptr: 0x00000000
    next-hop:   180.4.0.2

       TLU Channel: 2 Addr: 0x00712426
       ENTRY        0
        SW: 00000008 00000000 03e82000 20460b00
        HW: 00000008 00000000 03e82000 20460b00
       label1:        16002    label2:            16003
        label 3: 16009
       num of labels:     1    next ptr: 0x0020460b


  frr Flags        : 0x28c /* New flag to indicate that LFA is over PQ */
 Primary adjacency
 L2 Load info
  TLU Channel: 3 Addr: 0x20460b
  [HW: 0x00010000 0x00000000 0x00000000 0x30040b00]
    num. entries  : 1
    num. labels   : 0
    label 1       : 0
next ptr      : 0x30040b
 L2 Load Balancing Entry
  TLU Channel: 4 Addr: 0x30040b
    Entry[0]
    [HW: 0x00000004 0x00000084 0x01280440 0x00050000]
      dest. addr         : 0x4
      sponge queue       : 132
      egress port        : 0x128044
      rp destined        : no
      service destined   : no
      rp drop            : no
      hash type          : 0
      uidb index         : 0x5

 FRR backup info
  FRR Flags                  : 0x28c  /* updated to indicate PQ is active */
```

```
Cached backup adjacency
 Cached backup num. entries : 1
     protected num. entries : 1
     backup is a tunnel     : yes
     backup tunnel local label : 16007
     remote LFA active: 1
 Shared TLU Channel: 4 Addr: 0x300a02
   Entry[0]
   [HW: 0x00000004 0x00000086 0x01280480 0x00070000]
     dest. addr         : 0x4
     sponge queue       : 134
     egress port        : 0x128048
     rp destined        : no
     service destined   : no
     rp drop            : no
     hash type          : 0
     uidb index         : 0x7

   Load distribution: 0 (refcount 3)

   Hash  OK  Interface                Address
   0     Y   TenGigE0/2/0/4           180.4.0.2
```

Configuring IS-IS Overload Bit Avoidance

This task describes how to activate IS-IS overload bit avoidance.

Before you begin

The IS-IS overload bit avoidance feature is valid only on networks that support the following Cisco IOS XR features:

- MPLS
- IS-IS

SUMMARY STEPS

1. **configure**
2. **mpls traffic-eng path-selection ignore overload**

DETAILED STEPS

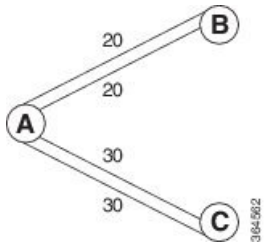
	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**mpls traffic-eng path-selection ignore overload** **Example:** `RP/0/RP0/CPU0:router(config)# mpls traffic-eng path-selection ignore overload`	Activates IS-IS overload bit avoidance.

ISIS Link Group

The ISIS Link-Group feature allows you to define a group or set of links, and raise or lower their ISIS metric according to a predefined number of active links.

When the total number of active links (in terms of ISIS adjacency) in a group falls below the configured number or members, a predefined offset is applied on the remaining active links. When the total number of active links in a group is reverted, ISIS restores the configured metric by removing the offset.

In the example below, Router A has to exit through router B and C. In between A and B there are two layer 3 links with the same ISIS metric (20). There is a similar setup between A and C (30). In normal operations, the traffic from A goes through B. If the ISIS Link-Group is not configured, even when the link between A and B fails, traffic is still routed through B. However, with ISIS Link-Group, you can set an offset of 20 with minimum-members of 2. Thus, if a link between A and B fails, the metric is raised to 40 (configured (20) + offset (20)), and so the traffic is routed to C. Further, you can define another ISIS Link-Group, this time between A and C. If a link between B and C fails, you can raise the offset to 20, and thus traffic is routed back to B.



Configure Link Group Profile

Perform this task to configure Intermediate System-to-Intermediate System (IS-IS) link group profiles:

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **link-group** *link-group-name* { **[metric-offset** *count* | **maximum**] | [**minimum-members** *count* | **revert-members** *count*] }
4. Use the **commit** or **end** command.
5. **show isis interface**
6. **show isis lsp**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **Example:**	Enters IS-IS configuration submode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config)# router isis purple`	
Step 3	**link-group** *link-group-name* { **[metric-offset** *count* | **maximum**] | [**minimum-members** *count* | **revert-members** *count*] }	Specifies link-group values. Following are the valid values: • **metric-offset**: Configures the metric offset for link group. The range is 1-16777214. The default metric offset range is between 1-63 for narrow metric; and 1-16777214 for wide metric. The **maximum** option here sets the maximum wide metric offset. All routers exclude this link from their SPF. • **minimum-members**: Configures the minimum number of members in the link group. The range is 2-64. • **revert-members**: Configures the number of members after which to revert in the link group. The range is 2-64. **Note** A link-group is only active after the **minimum-members** and **offset-metric** are configured in the profile. The **revert-members** is default to **minimum-members** if it is not configured.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**show isis interface** **Example:** `RP/0/RP0/CPU0:router# show isis interface`	(Optional) If link-group is configured on the interface, when showing the IS-IS interface-related topology, this command displays the link-group and current **offset-metric** value.
Step 6	**show isis lsp** **Example:** `RP/0/RP0/CPU0:router# show isis lsp`	(Optional) Displays the updated metric value.

Configure Link Group Profile: Example

The following is an example configuration, along with the show isis interface output:

```
router isis 1
 is-type level-2-only
 net 49.1111.0000.0000.0006.00
 link-group foo
  metric-offset 100
  revert-members 4
  minimum-members 2
 !
 address-family ipv4 unicast
  metric-style wide
 !
 interface GigabitEthernet0/0/0/1
  point-to-point
  address-family ipv4 unicast
   link-group foo

RP/0/RSP0/CPU0:Iguazu#sh isis interface gig 0/0/0/1
Thu Jun 11 14:55:32.565 CEST

GigabitEthernet0/0/0/1      Enabled
  Adjacency Formation:      Enabled
  Prefix Advertisement:     Enabled
  IPv4 BFD:                 Disabled
  IPv6 BFD:                 Disabled
  BFD Min Interval:         150
  BFD Multiplier:           3

  Circuit Type:             level-2-only (Interface circuit type is level-1-2)
  Media Type:               P2P
  Circuit Number:           0
  Extended Circuit Number:  36
  Next P2P IIH in:          8 s
  LSP Rexmit Queue Size:    0

  Level-2
    Adjacency Count:        1
    LSP Pacing Interval:    33 ms
    PSNP Entry Queue Size:  0

  CLNS I/O
    Protocol State:         Up
    MTU:                    1497
    SNPA:                   0026.9829.af19
    Layer-2 MCast Groups Membership:
      All ISs:              Yes

  IPv4 Unicast Topology:    Enabled
    Adjacency Formation:    Running
    Prefix Advertisement:   Running
    Metric (L1/L2):         110/110
    Weight (L1/L2):         0/0
    MPLS Max Label Stack:   1
    MPLS LDP Sync (L1/L2):  Disabled/Disabled
    Link-Group (L1/L2):     Configured/Configured
      Metric-Offset (L1/L2):100/100

  IPv4 Address Family:      Enabled
    Protocol State:         Up
    Forwarding Address(es): 100.5.6.6
```

```
  Global Prefix(es):      100.5.6.0/24

LSP transmit timer expires in 0 ms
LSP transmission is idle
Can send up to 9 back-to-back LSPs in the next 0 ms
```

Configure Link Group Interface

Perform this task to configure link group under Intermediate System-to-Intermediate System (IS-IS) interface and address-family sub-mode:

Note One IS-IS interface and address-family can specify only one link-group association. The default is for both levels regardless of the current circuit-type. The link-group association can be specified for one level only if configured.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **interface** *type interface-path-id*
4. **address-family ipv4** | **ipv6** [**unicast**]
5. **link-group** *link-group-name* [**level** {**1** | **2** }]
6. Use the **commit** or **end** command.
7. **show isis interface**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis purple`	Enters IS-IS configuration submode.
Step 3	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3`	Enters interface configuration mode.
Step 4	**address-family ipv4** | **ipv6** [**unicast**] **Example:**	Specifies the IPv6 address family and enters router address family configuration mode. • This example specifies the unicast IPv4 address family.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast`	
Step 5	**link-group** *link-group-name* [**level** {**1** | **2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# )#address-family ipv4 unicast link-group access level 1`	Specifies the link-group name and sets the tag at the level specified.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 7	**show isis interface** **Example:** `RP/0/RP0/CPU0:router# show  isis interface`	(Optional) If link-group is configured on the interface, when showing the IS-IS interface-related topology, this command displays the link-group value.

IS-IS Cost Fallback on IOS XR Bundle-Ether Interface

The IS-IS Cost Fallback on IOS XR Bundle-Ether Interface feature enables you to effectively manage the capacity of the network across the Bundle-Ether interfaces through a cost fallback mechanism. This mechanism increases the Bundle-Ether interface metric based on a active or total threshold. The threshold is a percentage of bandwidth.

Threshold = Bandwidth of the active bundle member links / the total bandwidth of all bundle links including both active and nonnative links.

The threshold based on the percentage of bandwidth makes it unnecessary to configure the changes as you upgrade the bandwidth.

Configuration Example

```
Router(config)# router isis 1
Router(config-isis)# is-type level-2-only
Router(config-isis)# net 47.0000.0000.0002.00
Router(config-isis)# log adjacency changes
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# exit
```

```
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# segment-routing mpls sr-prefer
Router(config-isis-af)# exit
Router(config-isis)# interface Bundle-Ether1
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric 6 level 1
Router(config-isis-af)# metric 21 level 2
Router(config-isis-af)# metric fallback bandwidth multiplier <1-100> threshold <1-100>
Router(config-isis-af)# exit
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# metric 30 level 1
Router(config-isis-af)# metric 60 level 2
Router(config-isis-af)# metric fallback bandwidth multiplier <factor; 20> threshold <#
bandwidth percentage; 80>
```

Running Configuration

```
router isis 1
 is-type level-2-only
 net 47.0000.0000.0002.00
 log adjacency changes
 address-family ipv4 unicast
 !
 address-family ipv6 unicast
  metric-style wide
 segment-routing mpls sr-prefer
 !
 interface Bundle-Ether1
  point-to-point
address-family ipv4 unicast
   metric 6 level 1
   metric 21 level 2
   metric fallback bandwidth multiplier <1-100> threshold <1-100>
  !
address-family ipv6 unicast
  metric 30 level 1
  metric 60 level 2
  metric fallback bandwidth multiplier <factor; 20> threshold <# bandwidth percentage; 80>
!
```

Verification

```
Router# show isis interface Bundle-Ether1
!!
  Bandwidth:                10000000
  Total bandwidth:          20000000
.
IPv4 Unicast Topology:    Enabled
    Adjacency Formation:    Running
    Prefix Advertisement:   Running
          Policy (L1/L2):   -/-
    Metric (L1/L2):         18/63
    Metric fallback:
      Bandwidth (L1/L2):    Active/Active
      Anomaly (L1/L2):      Inactive/Inactive
    Weight (L1/L2):         0/0
.
 IPv6 Unicast Topology:    Enabled
    Adjacency Formation:    Running
    Prefix Advertisement:   Running
          Policy (L1/L2):   -/-
```

```
    Metric (L1/L2):         600/1200
    Metric fallback:
      Bandwidth (L1/L2):    Active/Active
      Anomaly (L1/L2):      Inactive/Inactive
    Weight (L1/L2):         0/0
!!
```

The show output given below displays the information on IS-IS database.

```
Router# show isis database detail R2
Mon Jun  8 09:16:12.316 PDT

IS-IS 1 (Level-1) Link State Database
!!
  Metric: 600        MT (IPv6 Unicast) IPv6 10:0:13::/112
  Metric: 18         IP 10.0.13.0/24
  Total Level-1 LSP count: 1     Local Level-1 LSP count: 1
  IS-IS 1 (Level-2) Link State Database
  Metric: 1200       MT (IPv6 Unicast) IPv6 10:0:13::/112
  Metric: 63         IP 10.0.13.0/24
  Total Level-2 LSP count: 2     Local Level-2 LSP count: 1
!!
```

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 NCS 6000 Series 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	*Routing Command Reference for Cisco NCS 6000 Series Routers*
MPLS TE feature information	*Implementing MPLS Traffic Engineering on* module in *MPLS Configuration Guide for Cisco NCS 6000 Series Routers*
Bidirectional Forwarding Detection (BFD)	*Interface and Hardware Component Configuration Guide for Cisco NCS 6000 Series Routers* and *Interface and Hardware Component Command Reference for the Cisco NCS 6000 Series Routers*

Standards

Standards	Title
Draft-ietf-isis-ipv6-05.txt	*Routing IPv6 with IS-IS*, by Christian E. Hopps
Draft-ietf-isis-wg-multi-topology-06.txt	*M-ISIS: Multi Topology (MT) Routing in IS-IS*, by Tony Przygienda, Naiming Shen, and Nischal Sheth
Draft-ietf-isis-traffic-05.txt	*IS-IS Extensions for Traffic Engineering*, by Henk Smit and Toni Li
Draft-ietf-isis-restart-04.txt	*Restart Signaling for IS-IS*, by M. Shand and Les Ginsberg
Draft-ietf-isis-igp-p2p-over-lan-05.txt	*Point-to-point operation over LAN in link-state routing protocols*, by Naiming Shen
Draft-ietf-rtgwg-ipfrr-framework-06.txt	*IP Fast Reroute Framework*, by M. Shand and S. Bryant
Draft-ietf-rtgwg-lf-conv-frmwk-00.txt	*A Framework for Loop-free Convergence*, 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

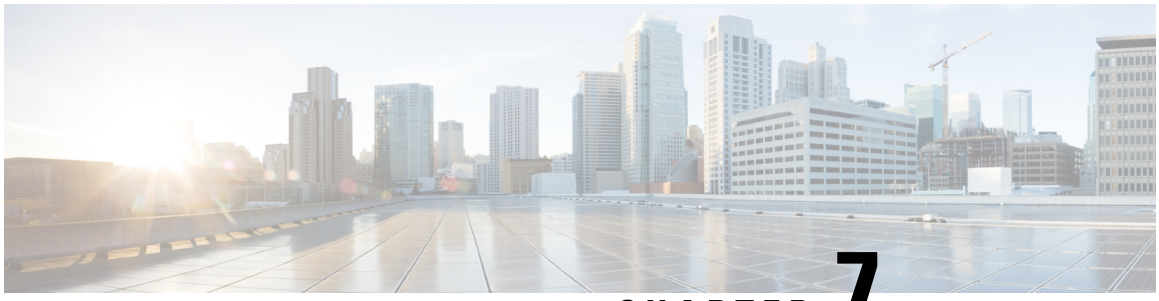
RFCs

RFCs	Title
RFC 1142	OSI IS-IS Intra-domain Routing Protocol

RFCs	Title
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
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
RFC 8405	Set SPF interval in IS-IS to postpone the IS-IS SPF computations

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 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 NCS 6000 Series 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 389 section of this module. To locate documentation for other commands that might appear during execution of a configuration task, search online in the

Feature History for Implementing OSPF

Release 5.0.0	This feature was introduced.

- Prerequisites for Implementing OSPF , on page 299
- Information About Implementing OSPF , on page 300
- How to Implement OSPF , on page 323
- Configuring IP Fast Reroute Loop-free Alternate, on page 370
- Configure Remote Loop-Free Alternate Paths for OSPF, on page 374
- Configuration Examples for Implementing OSPF , on page 382
- Where to Go Next, on page 389
- Additional References, on page 389

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* module of the *IP Addresses and Services Configuration Guide for Cisco NCS 6000 Series 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

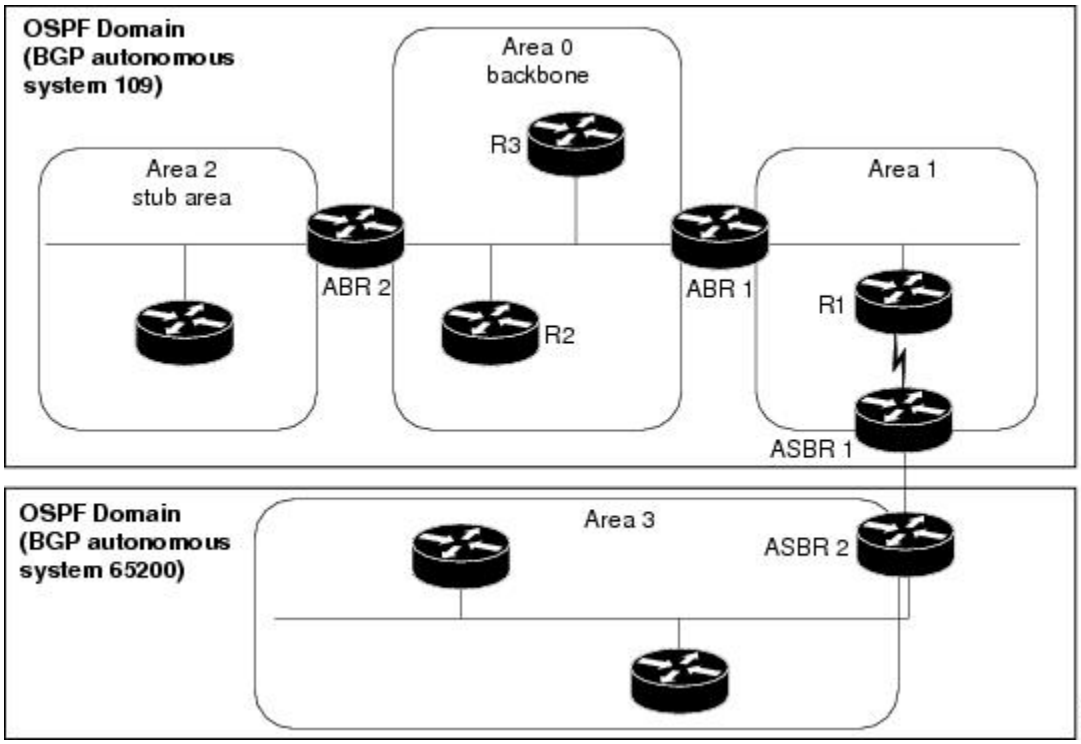
Use the default value of the command.

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 15: 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 15: OSPF Routing Components, on page 303 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 15: OSPF Routing Components, on page 303, 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 15: OSPF Routing Components, on page 303, 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 15: OSPF Routing Components, on page 303, 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 15: OSPF Routing Components, on page 303, 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 15: OSPF Routing Components, on page 303) 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.

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 319.

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.

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.

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

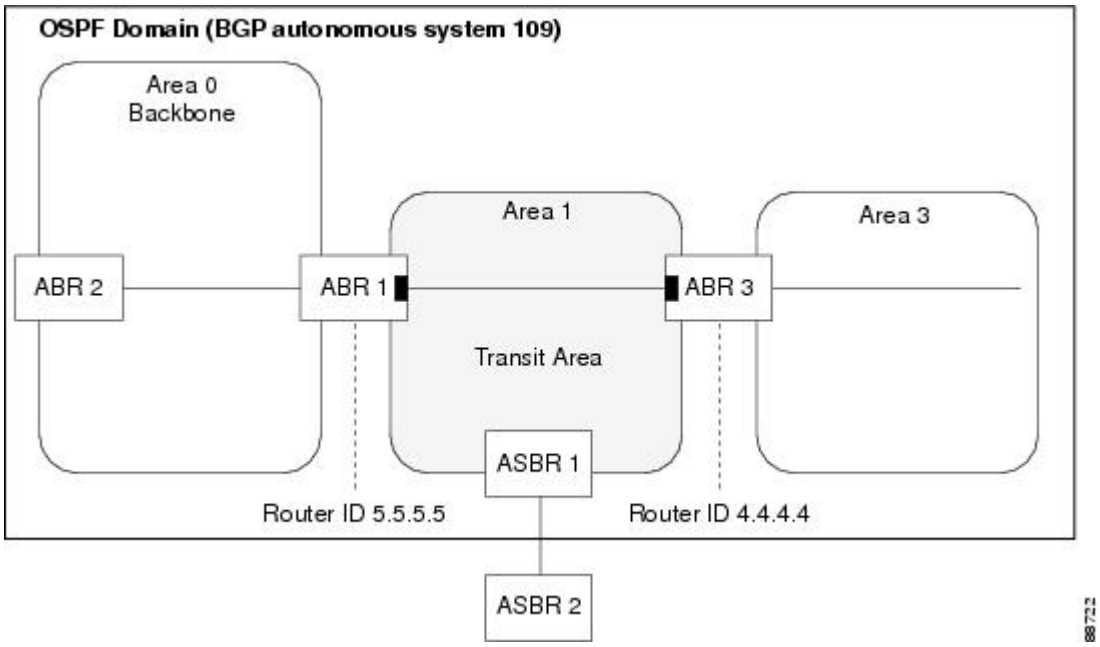
In OSPF, routing information from all areas is first summarized to the backbone area by ABRs. The same ABRs, in turn, propagate such received information to their attached areas. Such hierarchical distribution of routing information requires that all areas be connected to the backbone area (Area 0). 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 16: 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.

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 382 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.

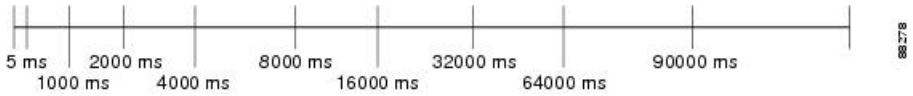
SPF calculations occur at the interval set by the **timers throttle spf** command. The wait interval indicates the amount of time to wait until the next SPF calculation occurs. Each wait interval after that calculation is twice as long as the previous interval until the interval reaches the maximum wait 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 17: 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.

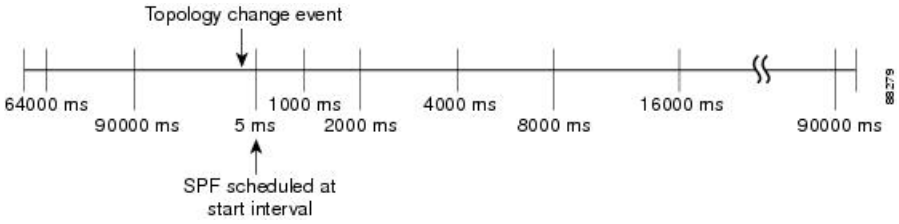


Notice that the wait interval between SPF calculations doubles when at least one topology change event is received during the previous wait interval. After the maximum wait time is reached, the wait interval remains the same until the topology stabilizes and no event is received in that interval.

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 18: Timer Intervals Reset After Topology Change Event, on page 313that a topology change event was received after the start of the maximum wait time interval and that the SPF intervals have been reset.

Figure 18: 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 352,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 316 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 OSFPv3 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.

Protocol Shutdown Mode

In this mode the OSPFv3 operation is completely disabled. This is accomplished by flushing self-originated link state advertisements (LSAs), immediately bringing down local OSPFv3-supported interfaces, and clearing the Link State Database (LSDB). The non-local LSDB entries are removed by OSPFv3, These are not flooded (MaxAged).

The protocol shutdown mode can be invoked either manually through the **protocol shutdown** command that disables the protocol instance or when the OSPFv3 process runs out of memory. These events occur when protocol shut down is performed:

- The local Router LSA and all local Link LSAs are flushed. All other LSAs are eventually aged out by other OSPFv3 routers in the domain.
- OSPFv3 neighbors not yet in Full state with the local router are brought down with the Kill_Nbr event.
- After a three second delay, empty Hello packets are immediately sent to each neighbor that has an active adjacency.
  - An empty Hello packet is sent periodically until the dead_interval has elapsed.
  - When the dead_interval elapses, Hello packets are no longer sent.

After a Dead Hello interval delay (4 X Hello Interval), the following events are then performed:

- The LSA database from that OSPFv3 instance is cleared.
- All routes from RIB that were installed by OSPFv3 are purged.

The router will not respond to any OSPF control packets it receives from neighbors while in protocol shutdown state.

Protocol Restoration

The method of restoring the protocol is dependent on the trigger that originally invoked the shut down. If the OSPFv3 was shut down using the **protocol shutdown** command, then use the **no protocol shutdown** command to restore OSPFv3 back to normal operation. If the OSPFv3 was shutdown due to a Critical Memory message from the sysmon, then a Normal Memory message from sysmon, which indicates that sufficient memory has been restored to the processor, restores the OSPFv3 protocol to resume normal operation. When OSPFv3 is shutdown due to the Critical Memory trigger, it must be manually restarted when normal memory levels are restored on the route processor. It will not automatically restore itself.

These events occur when the OSPFv3 is restored:

1. All OSPFv3 interfaces are brought back up using the Hello packets and database exchange.
2. The local router and link LSAs are rebuilt and advertised.
3. The router replies normally to all OSPFv3 control messages received from neighbors.
4. Routes learned from other OSPFv3 routers are installed in RIB.

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 then 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.

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 64 and the default number of maximum paths is 64.

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 OSF speakers are attached. In the case of native broadcast networks, the interface must be configured as an OPSF 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 theshold
  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.

LDP IGP auto-configuration can also be explicitly disabled on an individual interface basis under LDP using the **igp auto-config disable** command. This allows LDP to receive all OSPF interfaces minus the ones explicitly disabled.

See *MPLS Configuration Guide for Cisco NCS 6000 Series 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 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* module of the *MPLS Configuration Guide for Cisco NCS 6000 Series 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.

Multiple OSPFv3 Instances

SNMPv3 supports "contexts" that can be used to implement MIB views on multiple OSPFv3 instances, in the same system.

IGP link state

IGP Link-State Database Distribution

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

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

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

OSPFv2 Unequal Cost Load Balancing

Unequal Cost Load Balancing feature in Cisco IOS XR OSPFv2 feature enables Unequal Cost Multipath (UCMP) calculation based on configured prefix-list and based on variance factor. UCMP path can be calculated for all prefixes or only for selected prefixes based on the configuration. Selected interfaces can be excluded to be used as a candidate for UCMP paths. The calculated UCMP paths are then installed in the routing information base (RIB) subject to the max-path limit.

The OSPFv2 interior gateway protocol is used to calculate paths to prefixes inside an autonomous system. OSPF calculates up to maximum paths (max-path) equal cost multi-paths (ECMPs) for each prefix, where max-path is either limited by the router support or is configured by the user.

UCMP Paths Calculation

In some topologies, alternate paths to prefix exist even though their metric is higher then the metric of the best path(s). These paths are called Unequal Cost Multipaths (UCMPs). These paths are guaranteed to be loop free. Users can send some portion of the traffic down these paths to better utilize the available bandwidth. However, the UCMP paths are not discovered by the traditional Dijkstra calculation. Additional computation is required to discover these paths.

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

More than 32 ECMP and UCMP paths are not supported for these features:

- LI
- GRE
- BVI
- NetFlow
- Satellite
- MCAST
- SPAN
- PWHE
- ABF
- P2MP

- MVPN
- VPLS
- L2TPv3
- LISP
- VIDMON
- PBR

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

	Command or Action	Purpose

Step 2	Do one of the following:
	• **router ospf** *process-name*
	• **router ospfv3** *process-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		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 *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**router-id** { *router-id* } **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 IP address as the router ID.
Step 4	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Enters area configuration mode and configures an area for the OSPF process. • Backbone areas have an area ID of 0. • Nonbackbone areas have a nonzero area ID. • The *area-id* 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** *type interface-path-id* **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. • 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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.
9. Repeat this task on all other routers in the stub area or NSSA.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	Do one of the following: • **router ospf** *process-name*	Enables OSPF routing for the specified routing process and places the router in router configuration mode.

	Command or Action	Purpose

	• **router ospfv3** *process-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		or
		Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.
		Note The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**router-id** { *router-id* } **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 IP address as the router ID.
Step 4	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 1`	Enters area configuration mode and configures a nonbackbone area for the OSPF process. • The *area-id* 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:
	• **stub** [**no-summary**]
	• **nssa** [**no-redistribution**] [**default-information-originate**] [**no-summary**] [**translate**] [**translate always**]
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# stub no summary`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar)# nssa no-redistribution`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar)# nssa translate <type number> always`
		Defines the nonbackbone area as a stub area.
		• 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.
		or
		Defines an area as an NSSA.

Step 6	Do one of the following: • **stub** • **nssa** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# stub`	(Optional) Turns off the options configured for stub and NSSA areas. • 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**

	Command or Action	Purpose
	or `RP/0/RP0/CPU0:router(config-ospf-ar)# 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	**default-cost** *cost* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)#default-cost 15`	(Optional) Specifies a cost for the default summary route sent into a stub area or an NSSA. • 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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 9	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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

Step 2	Do one of the following:
	• **router ospf** *process-name*
	• **router ospfv3** *process-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		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 *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**router-id** { *router-id* } **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 IP address as the router ID.
Step 4	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Enters area configuration mode and configures an area for the OSPF process. • The example configures a backbone area. • The *area-id* 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

	Command or Action	Purpose
		one form or the other for an area. We recommend using the IPv4 address notation.
Step 5	**network** { **broadcast** | **non-broadcast** | { **point-to-multipoint** [**non-broadcast**] | **point-to-point** }} **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# network non-broadcast`	Configures the OSPF network type to a type other than the default for a given medium. • The example sets the network type to NBMA.
Step 6	**dead-interval** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# dead-interval 40`	(Optional) Sets the time to wait for a hello packet from a neighbor before declaring the neighbor down.
Step 7	**hello-interval** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# hello-interval 10`	(Optional) Specifies the interval between hello packets that OSPF sends on the interface. **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 then default value.
Step 8	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/2/0/0`	Enters interface configuration mode and associates one or more interfaces for the area configured in Step 4. • 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	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**]]
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# neighbor 10.20.20.1 priority 3 poll-interval 15`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# neighbor fe80::3203:a0ff:fe9d:f3fe`
		Configures the IPv4 address of OSPF neighbors interconnecting to nonbroadcast networks.
		or
		Configures the link-local IPv6 address of OSPFv3 neighbors.
		• The *ipv6-link-local-address* 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 priority value should match the actual priority setting on the neighbor router. The neighbor priority default

	Command or Action	Purpose
		value is zero. This keyword does not apply to point-to-multipoint interfaces. • 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	exit **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# exit`	Enters area configuration mode.
Step 12	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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 #unique_417 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 306 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router 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. **Note** The *process-name* argument is any alphanumeric string no longer than 40 characters.

	Command or Action	Purpose
Step 3	**router-id** { *router-id* } **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.
Step 5	**message-digest-key** *key-id* **md5** { *key* | **clear** *key* | **encrypted** *key* | **LINE**} **Example:** `RP/0/RP0/CPU0:router(config-ospf)#message-digest-key 4 md5 yourkey`	Specifies the MD5 authentication key for the OSPF process. • The neighbor routers must have the same key identifier.
Step 6	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Enters area configuration mode and configures a backbone area for the OSPF process.
Step 7	**interface** *type interface-path-id* **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 backbone area. • All interfaces inherit the authentication parameter values specified for the OSPF process (Step 4, Step 5, and Step 6).
Step 8	Repeat Step 7 for each interface that must communicate, using the same authentication.	—
Step 9	**exit** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# exit`	Enters area OSPF configuration mode.
Step 10	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 1`	Enters area configuration mode and configures a nonbackbone area 1 for the OSPF process. • The *area-id* 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 11	**authentication** [**message-digest** | **null**] **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# authentication`	Enables Type 1 (plain text) authentication that provides no security. • 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	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/1/0/0`	Enters interface configuration mode and associates one or more interfaces to the nonbackbone area 1 specified in Step 7. • All interfaces configured inherit the authentication parameter values configured for area 1.
Step 13	Repeat Step 12 for each interface that must communicate, using the same authentication.	—
Step 14	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/3/0/0`	Enters interface configuration mode and associates one or more interfaces to a different authentication type.
Step 15	**authentication** [**message-digest** | **null**] **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# authentication null`	Specifies no authentication on GigabitEthernet interface 0/3/0/0, overriding the plain text authentication specified for area 1. • By default, all of the interfaces configured in the same area inherit the same authentication parameter values of the area.
Step 16	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

Step 2	Do one of the following:
	• **router ospf** *process-name*
	• **router ospfv3** *process-name*
	Example:
	`RP/0/RP0/CPU0:router:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		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 *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**router-id** { *router-id* } **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 IP address as the router ID.
Step 4	Perform Step 5 or Step 6 or both to control the frequency that the same LSA is originated or accepted.	—

	Command or Action	Purpose
Step 5	**timers lsa refresh** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# timers lsa refresh 1800`	Sets how often self-originated LSAs should be refreshed, in seconds. • The default is 1800 seconds for both OSPF and OSPFv3.
Step 6	**timers lsa min-arrival** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# timers lsa min-arrival 2`	Limits the frequency that new processes of any particular OSPF Version 2 LSA can be accepted during flooding. • The default is 1 second.
Step 7	**timers lsa group-pacing** *seconds* **Example:** `RP/0/ /CPU0:router(config-ospf)# timers lsa group-pacing 1000`	Changes the interval at which OSPF link-state LSAs are collected into a group for flooding. • The default is 240 seconds.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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 309 .

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 388.

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 305 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* module in *Routing Command Reference for Cisco NCS 6000 Series 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. Use the **commit** or **end** command.
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:
	• **show ospf** [*process-name*]
	• **show ospfv3** [*process-name*]
	Example:
	`RP/0//CPU0:router# show ospf`
	or
	`RP/0//CPU0:router# show ospfv3`
		(Optional) Displays general information about OSPF routing processes.
		• 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** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

Step 3	Do one of the following:
	• **router ospf** *process-name*
	• **router ospfv3** *process-name*
	Example:
	`RP/0//CPU0:router(config)# router ospf 1`
	or
	`RP/0//CPU0:router(config)# router ospfv3 1`
		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 *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 4	**router-id** { *router-id* } **Example:** `RP/0//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 5	**area** *area-id* **Example:** `RP/0//CPU0:router(config-ospf)# area 1`	Enters area configuration mode and configures a nonbackbone area for the OSPF process. • The *area-id* 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** *router-id* **Example:** `RRP/0//CPU0:router(config-ospf-ar)# virtual-link 10.3.4.5`	Defines an OSPF virtual link. • See .

	Command or Action	Purpose
Step 7	**authentication message-digest** **Example:** `RP/0//CPU0:router(config-ospf-ar-vl)#authentication message-digest`	Selects MD5 authentication for this virtual link.
Step 8	**message-digest-key** *key-id* **md5** { *key* | **clear** *key* | **encrypted** *key* } **Example:** `RP/0//CPU0:router(config-ospf-ar-vl)#message-digest-key 4 md5 yourkey`	Defines an OSPF virtual link. • See to understand a virtual link. • The *key-id* argument is a number in the range from 1 to 255. The *key* 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** *key* 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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 11	Do one of the following:
	• **show ospf** [*process-name*] [*area-id*] **virtual-links**
	• **show ospfv3** [*process-name*] **virtual-links**
	Example:
	`RP/0//CPU0:router# show ospf 1 2 virtual-links`
	or
	`RP/0//CPU0:router# show ospfv3 1 virtual-links`
		(Optional) Displays the parameters and the current state of OSPF virtual links.

Examples

In the following example, the **show ospfv3 virtual links** XR EXEC 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

Step 2	Do one of the following:
	• **router ospf** *process-name*
	• **router ospfv3** *process-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		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 *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**router-id** { *router-id* } **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** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area`	Enters area configuration mode and configures a nonbackbone area for the OSPF process. • The *area-id* 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: • **range** *ip-address mask* [**advertise** | **not-advertise**] • **range** *ipv6-prefix / prefix-length* [**advertise** | **not-advertise**] **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# range 192.168.0.0 255.255.0.0 advertise` or	Consolidates and summarizes OSPF routes at an area boundary. • 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.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-ospf-ar)# range 4004:f000::/32 advertise`	• 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** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/2/0/3`	Enters interface configuration mode and associates one or more interfaces to the area.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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* module in the *Routing Configuration Guide for Cisco NCS 6000 Series 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

Step 2	Do one of the following:
	• **router ospf** *process-name*
	• **router ospfv3** *process-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		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 *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**router-id** { *router-id* } **Example:** `RRP/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	**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*]
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# redistribute bgp 100`
	or
	`RP/0/RP0/CPU0:router(config-router)#redistribute bgp 110`
		Redistributes OSPF routes from one routing domain to another routing domain.
		or
		Redistributes OSPFv3 routes from one routing domain to another routing domain.
		• 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 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

	Command or Action	Purpose
		link type associated with the default route advertised into the OSPFv3 routing domain is the Type 1 external route. **Note** RPL is not supported for OSPFv3.

Step 5	Do one of the following:
	• **summary-prefix** *address mask* [**not-advertise**] [**tag** *tag*]
	• **summary-prefix** *ipv6-prefix / prefix-length* [**not-advertise**] [**tag** *tag*]
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# summary-prefix 10.1.0.0 255.255.0.0`
	or
	`RP/0/RP0/CPU0:router(config-router)# summary-prefix 2010:11:22::/32`
		(Optional) Creates aggregate addresses for OSPF.
		or
		(Optional) Creates aggregate addresses for OSPFv3.
		• 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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.
8. Do one of the following:
   - **show ospf** [*process-name*]
   - **show ospfv3** [*process-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

Step 2	Do one of the following:
	• **router ospf** *process-name*
	• **router ospfv3** *process-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		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 *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**router-id** { *router-id* } **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	**timers throttle spf** *spf-start spf-hold spf-max-wait* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# timers throttle spf 10 4800 90000`	Sets SPF throttling timers.
Step 5	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)#  area 0`	Enters area configuration mode and configures a backbone area. • The *area-id* 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** *type interface-path-id* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 8	Do one of the following:
	• **show ospf** [*process-name*]
	• **show ospfv3** [*process-name*]
	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** XR EXEC 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 msecs
  Minimum hold time between two consecutive SPFs 100 msecs
  Maximum wait time between two consecutive SPFs 1000 msecs
  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* module in *Routing Command Reference for Cisco NCS 6000 Series 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router 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. **Note** The *process-name* argument is any alphanumeric string no longer than 40 characters.
Step 3	**router-id** { *router-id* } **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	Do one of the following: • **nsf cisco** • **nsf cisco enforce global** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# nsf cisco enforce global`	Enables Cisco NSF operations for the OSPF process. • 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.

	Command or Action	Purpose
Step 5	**nsf interval** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# nsf interval 120`	Sets the minimum time between NSF restart attempts. **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.
Step 6	**nsfflush-delay-time***seconds* **Example:** `RP/0/RP0/CPU0:router(config-ospf)#nsf flush-delay-time 1000`	Sets the maximum time allowed for external route learning in seconds.
Step 7	**nsflifetime***seconds* **Example:** `RP/0/RP0/CPU0:router(config-ospf)#nsf lifetime 90`	Sets the maximum route lifetime of NSF following a restart in seconds.
Step 8	**nsfietf** **Example:** `RP/0/RP0/CPU0:router(config-ospf)#nsf ietf`	Enables ietf graceful restart.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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 module of the *MPLS Configuration Guide for Cisco NCS 6000 Series 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. Use the **commit** or **end** command.
9. **show ospf** [*process-name*] [*area-id*] **mpls traffic-eng** { **link** | **fragment** }

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router 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. **Note** The *process-name* argument is any alphanumeric string no longer than 40 characters.
Step 3	**router-id** { *router-id* } **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	**mpls traffic-eng router-id** *interface-type interface-instance* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# mpls traffic-eng router-id loopback 0`	(Optional) Specifies that the traffic engineering router identifier for the node is the IP address associated with a given interface. • 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

	Command or Action	Purpose
		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** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Enters area configuration mode and configures an area for the OSPF process. • The *area-id* 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:** `RP/0/RP0/CPU0:router(config-ospf)# mpls traffic-eng`	Configures the MPLS TE under the OSPF area.
Step 7	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface interface loopback0`	Enters interface configuration mode and associates one or more interfaces to the area.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 9	**show ospf** [*process-name*] [*area-id*] **mpls traffic-eng** { **link** | **fragment** } **Example:** `RP/0/RP0/CPU0:router# show ospf 1 0 mpls traffic-eng link`	(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** XR EXEC 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** XR EXEC 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** XR EXEC 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** XR EXEC 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. Use the **commit** or **end** command.
8. **show ospfv3** [*process-name* [*area-id*]] **database grace**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router ospfv3** *process-name* **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. • The default lifetime is 95 seconds. • The range is 90 to 3600 seconds.
Step 5	**graceful-restart interval** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-ospfv3)# graceful-restart interval 120`	Specifies the interval (minimal time) between graceful restarts on the current router. • 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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

	Command or Action	Purpose
Step 8	**show ospfv3** [*process-name* [*area-id*]] **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 198.51.100.1
 Initial SPF schedule delay 5000 msecs
 Minimum hold time between two consecutive SPFs 10000 msecs
 Maximum wait time between two consecutive SPFs 10000 msecs
 Initial LSA throttle delay 0 msecs
 Minimum hold time for LSA throttle 5000 msecs
 Maximum wait time for LSA throttle 5000 msecs
 Minimum LSA arrival 1000 msecs
 LSA group pacing timer 240 secs
 Interface flood pacing timer 33 msecs
 Retransmission pacing timer 66 msecs
 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 (192.0.2.1) (Process ID 1)

            Grace (Type-11) Link States (Area 0)

LS age: 2
```

```
LS Type: Grace Links
Link State ID: 34
Advertising Router: 192.0.2.1
LS Seq Number: 80000001
Checksum: 0x7a4a
Length: 36
    Grace Period : 90
    Graceful Restart Reason : Software reload/upgrade
```

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. Use the **commit** or **end** command.
7. **show rpl route-policy** *route-policy name* **detail**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

Step 2	**prefix-set** *prefix-set name*
	Example:
	`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`
		Configures the prefix set.

	Command or Action	Purpose

Step 3	**route-policy** *route-policy name* **if destination in** *prefix-set name* **then set spf-priority** {**critical** | **high** | **medium**} **endif**
	Example:
	`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`
		Configures route policy and sets OSPF SPF priority.

Step 4	Use one of these commands:
	• **router ospf** *ospf-name*
	• **router ospfv3** *ospfv3-name*
	Example:
	`RP/0/RP0/CPU0:router# router ospf 1`
	Or
	`RP/0/RP0/CPU0:router# router ospfv3 1`
		Enters Router OSPF configuration mode.

Step 5	**spf prefix-priority route-policy** *route-policy name* **Example:** Or `RP/0/RP0/CPU0:router(config-ospfv3)#spf prefix-priority route-policy ospf3-spf-priority`	Configures SPF prefix-priority for the defined route policy. **Note** Configure the **spf prefix-priority** command under router OSPF.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 7	**show rpl route-policy** *route-policy name* **detail**
	Example:
	`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`
		Displays the set SPF prefix priority.

	Command or Action	Purpose

	``
	!
	route-policy ospf-spf-priority
	  if destination in ospf-critical-prefixes then
	  set spf-priority critical
	  endif
	end-policy
	!
	``	

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 OSF speakers are attached. In the case of native broadcast networks, the interface must be configured as an OPSF 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router 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. **Note** The *process-name* argument is any alphanumeric string no longer than 40 characters.
Step 3	**area** *area-id* **Example:**	Enters area configuration mode and configures a backbone area.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-ospf)# area 0`	• The *area-id* 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** *type interface-path-id* **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** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 1`	Enters area configuration mode and configures an area used for multiple area adjacency. • The *area-id* 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** *type interface-path-id* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router 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. **Note** The *process-name* argument is any alphanumeric string no longer than 40 characters.
Step 3	**mpls ldp auto-config** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# mpls ldp auto-config`	Enables LDP IGP interface auto-configuration for an OSPF instance. • Optionally, this command can be configured for an area of an OSPF instance.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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 IGPs.

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** *process-name* **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: • **mpls ldp sync** • **area** *area-id* **mpls ldp sync** • **area** *area-id* **interface** *name* **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* module of the *System Security Configuration Guide for Cisco NCS 6000 Series 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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router 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. **Note** The *process-name* argument is any alphanumeric string no longer than 40 characters.
Step 3	**router-id** { *router-id* } **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** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 1`	Enters area configuration mode. The *area-id* 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** *type interface-path-id* **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** *keychain* | **null**] **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# authentication message-digest keychain ospf_int1`	Configures an MD5 keychain. **Note** In the example, the *ospf_intl* keychain must be configured before you attempt this step.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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. Use the **commit** or **end** command.
11. **show ospf** [*process-name*] [*area-id*] **interface** [*type interface-path-id*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router 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. **Note** The *process-name* argument is any alphanumeric string no longer than 40 characters.

	Command or Action	Purpose
Step 3	**router-id** { *router-id* } **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. • 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** *spf-start spf-hold spf-max-wait* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# timers throttle spf 500 500 10000`	(Optional) Sets SPF throttling timers.
Step 7	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 1`	Enters area configuration mode. The *area-id* 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** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEternet0/5/0/0`	Enters interface configuration mode and associates one or more interfaces to the area.
Step 9	**security ttl** [**disable** | **hops** *hop-count*] **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# security ttl hopes 2`	Sets the security TTL value in the IP header for OSPF packets.
Step 10	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 11	**show ospf** [*process-name*] [*area-id*] **interface** [*type interface-path-id*] **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*] [*area-id*] **interface** [*type interface-path-id*]
6. **show** { **ospf** | **ospfv3** }[*process-name*] [*area-id*] **neighbor** [*t ype 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*] **statistics** [**neighbor** [*type interface-path-id*] [*ip-address*]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show** { **ospf**	**ospfv3** } [*process-name*] **Example:** `RP/0/RP0/CPU0:router# show ospf group1`	(Optional) Displays general information about OSPF routing processes.
Step 2	**show** { **ospf**	**ospfv3** } [*process-name*] **border-routers** [*router-id*] **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** } [*process-name*] **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. • The various forms of this command deliver information about different OSPF LSAs.
Step 4	**show** { **ospf**	**ospfv3** } [*process-name*] [*area-id*] **flood-list interface** *type interface-path-id* **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** } [*process-name*] [*area-id*] **interface** [*type interface-path-id*] **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** }[*process-name*] [*area-id*] **neighbor** [*t ype interface- path-id*] [*neighbor-id*] [**detail**] **Example:**	(Optional) Displays OSPF neighbor information on an individual interface basis.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# show ospf 100 neighbor`	
Step 7	**clear** { **ospf** | **ospfv3** }[*process-name*] **process** **Example:** `RP/0/ /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*] **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** **Example:**	Enters XR Config mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# configure`	
Step 2	**router ospf** *process-name* **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 *process-name* argument is any alphanumeric string no longer than 40 characters.
Step 3	**queue dispatch incoming** *count* **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** *count* **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** *count* **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** } *count* **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.

IGP link state

IGP Link-State Database Distribution

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

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

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

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router ospf** *process-name* **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** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)#area 1`	Enters area configuration mode.
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)#  interface GigabitEternet0/5/0/0`	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.

	Command or Action	Purpose
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router ospf** *process-name* **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** *area-id* **Example:** `RP/0/RP0/CPU0:router(config)#area area-id`	Enters area configuration mode.
Step 4	**interface** *type interface-path-id* **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.

	Command or Action	Purpose
Step 5	**fast-reroute per-link exclude interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# fast-reroute per-link exclude interface GigabitEternet0/5/0/1`	Excludes an interface from IP fast reroute per-link computation.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Enabling 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** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router ospf** *instance-id* **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.
Step 3	**segment-routing mpls**	

	Command or Action	Purpose
	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-mapadvertise-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-preferprefix-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
!
```

Configure Remote Loop-Free Alternate Paths for OSPF

With Loop-Free Alternate (LFA) paths only the immediate next hops (directly-connected neighbors) are used as backups to a destination. Though this works well in most topologies, it fails in ring topologies, because the immediate next hop is likely to use the source router itself to forward traffic, thereby creating loops.

As a result, during link failures, extended LFA or Remote LFA (RLFA) is required to enable the router to use a non-directly connected next hop as backup. RLFA ensures that this nearest non-connected neighbor does not loop back to the source router.

RLFA uses an LDP tunnel between source and backup routers and is implemented in IPv4, IPv6, and MPLS networks.

Remote LFA Process

The remote LFA process on the source router involves:

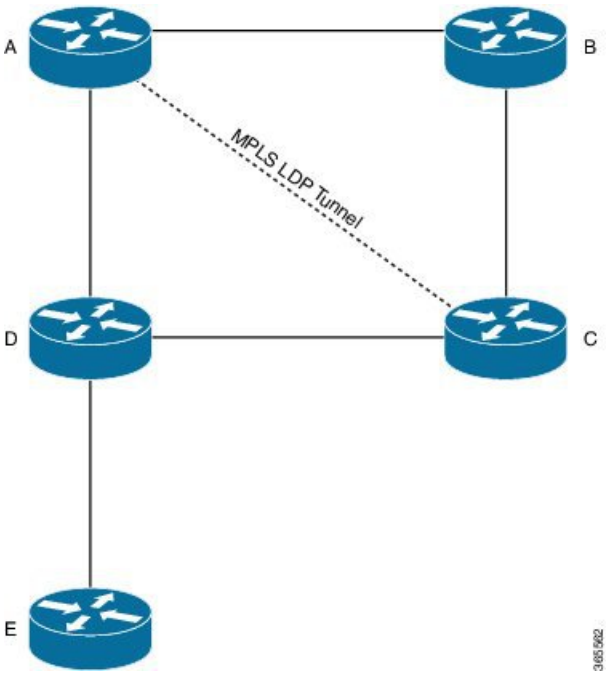
1. For a protected link between the source router and the immediate next hop, the IGP (OSPF or IS-IS) computes the remote LFA backup path by using the PQ algorithm.
2. The IGP updates the RIB table with the remote LFA path information.
3. LDP establishes a session with the remote router to exchange labels for prefixes.
4. LDP sets up MPLS forwarding for the protected prefix, and the corresponding backup path.
5. On link failure, and fast reroute trigger, the remote LFA backup path is activated with less than 50 millisecond convergence time.
6. The backup path is active until the IGP converges to the new primary path.

Remote LFA Topology

Consider the topology in the following figure. The best path from Source Router A to Destination Router E is through Router D. The best backup path for this route would be: Router A -> B -> C -> D -> Router E. Because Router B uses Router A as its primary next hop, this backup path cannot be used and LFA fails in this topology.

As a result, by implementing RLFA, Router A calculates its nearest non-directly connected neighbor that does not route back to it. In this example, Router A chooses Router C as its RLFA backup, and uses LDP to establish a tunnel between them. Traffic from Router A is routed to Router C on fast reroute trigger.

Figure 19: Remote LFA Topology



Configuration

To configure RLFA with OSPF, use the following steps.

1. Configure the interface(s) of the router.

```
RP/0/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#interface gigabitEthernet 0/0/0/1
RP/0/RP0/CPU0:router(config-if)#ipv4 address 10.1.1.1 255.255.255.0
RP/0/RP0/CPU0:router(config-if)#no shutdown
RP/0/RP0/CPU0:router(config-if)#exit
RP/0/RP0/CPU0:router(config)#interface gigabitEthernet 0/0/0/2
RP/0/RP0/CPU0:router(config-if)#ipv4 address 10.10.10.1 255.255.255.0
RP/0/RP0/CPU0:router(config-if)#no shutdown
RP/0/RP0/CPU0:router(config-if)#exit
RP/0/RP0/CPU0:router(config)#interface loopback0
RP/0/RP0/CPU0:router(config-if)#ipv4 address 10.10.10.1 255.255.255.255
RP/0/RP0/CPU0:router(config-if)#no shutdown
RP/0/RP0/CPU0:router(config-if)#exit
```

2. Configure OSPF.

```
RP/0/RP0/CPU0:router(config)#router ospf rlfa
RP/0/RP0/CPU0:router(config-ospf)#router-id 10.1.1.1
RP/0/RP0/CPU0:router(config-ospf)#area 1
```

3. Add the configured interface(s) to OSPF, and configure RLFA.

```
RP/0/RP0/CPU0:router(config-ospf-ar)#interface gigabitEthernet 0/0/0/1
RP/0/RP0/CPU0:router(config-ospf-ar-if)#fast-reroute per-prefix remote-lfa tunnel mpls-ldp

RP/0/RP0/CPU0:router(config-ospf-ar-if)#fast-reroute per-prefix remote-lfa maximum-cost
 20
RP/0/RP0/CPU0:router(config-ospf-ar-if)#exit
```

```
RP/0/RP0/CPU0:router(config-ospf-ar)#exit
RP/0/RP0/CPU0:router(config-ospf)#microloop avoidance protected
RP/0/RP0/CPU0:router(config-ospf)#exit
```

The **maximum-cost** command is required to limit the range of remote LFAs. The **microloop avoidance** command is used to delay the convergence of all or protected prefixes (as configured in this example). For more information on these commands, see the *Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference*.

4. Commit your configuration.

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

5. Confirm your configuration.

```
RP/0/RP0/CPU0:router(config)#show running-config
...
!
interface GigabitEthernet0/0/0/1
 ipv4 address 10.1.1.1 255.255.255.0
!
interface GigabitEthernet0/0/0/2
 ipv4 address 10.10.10.1 255.255.255.0
 shutdown

interface loopback0
 ipv4 address 10.1.1.1 255.255.255.255
 shutdown
...
!
router ospf rlfa
 router-id 10.1.1.1
 microloop avoidance protected
 area 1
  interface GigabitEthernet0/0/0/1
   fast-reroute per-prefix remote-lfa tunnel mpls-ldp
   fast-reroute per-prefix remote-lfa maximum-cost 20
  !
 !
!
...
```

Sample Verification Outputs

You can run the show commands mentioned in this section to verify whether RLFA is operational in your network. This section lists the sample outputs that are retrieved depending on your network topology.

RIB Verification

Verify the presence of the remote backup paths in the RIB.

```
RP/0/RP0/CPU0:router#show route 10.1.1.1/32

Routing entry for 10.1.1.1/32
  Known via "ospf 100", distance 110, metric 20, type extern 2
  Installed Sep  8 15:18:33.214 for 2d00h
  Routing Descriptor Blocks
    13.0.0.3, from 10.1.1.1, via GigabitEthernet0/0/0/1, Protected
    ...
    131.1.1.4, from 10.10.10.1 via GigabitEthernet0/0/0/2, Backup (remote)
    ...
```

```
RP/0/RP0/CPU0:router#show route 10.1.1.1/32 detail

Routing entry for 10.1.1.1/32
  Known via "ospf 100", distance 110, metric 20, type extern 2
  Installed Sep  8 15:18:33.214 for 2d00h
  Routing Descriptor Blocks
    13.0.0.3, from 10.1.1.1, via GigabitEthernet0/0/0/1, Protected
      Path id: 1     Path ref count: 0
      Backup path id: 33
    ...
    131.1.1.4, from 10.10.10.1 via GigabitEthernet0/0/0/2, Backup (remote)
     Remote LFA is 4.4.4.4
      Path id: 33   Path refcount:1
   ...
```

OSPF Verification

Verify the presence of the remote backup paths in the OSPF topology table.

```
RP/0/RP0/CPU0:router#show ospf rlfa routes backup-path
...

Topology Table for ospf rlfa with ID 10.1.1.1

Codes: O - Intra area, O IA - Inter area
       O E1 - External type 1, O E2 - External type 2
       O N1 - NSSA external type 1, O N2 - NSSA external type 2

O    10.3.10.0/24, metric 1
       10.3.10.143, directly connected, via GigabitEthernet0/0/0/1
O    10.3.11.0/24, metric 1
       10.3.11.143, directly connected, via GigabitEthernet0/0/0/2
O    192.168.0.145/32, metric 2

       10.3.10.145, from 192.168.0.145, via GigabitEthernet0/0/0/1, path-id 2
           Backup path: Remote, LFA: 11.0.0.1
             10.3.11.145, from 192.168.0.145, via GigabitEthernet0/0/0/2, protected bitmap
 0x2
              Attributes: Metric: 0,
       10.3.11.145, from 192.168.0.145, via GigabitEthernet0/0/0/2, path-id 1
           Backup path: Remote, LFA: 11.0.0.2
             10.3.10.145, from 192.168.0.145, via GigabitEthernet0/0/0/1, protected bitmap
 0x1
              Attributes: Metric: 0,
```

FIB Verification

Verify the presence of remote backup paths in the FIB.

```
RP/0/RP0/CPU0:router# show cef 10.1.1.1 detail

10.1.1.1/32, version 6, internal 0x4004001 (ptr 0x1d5977f4) [1], 0x0 (0x1d563344), 0x450
(0x1dbab050)
 Updated Apr 27 13:36:03.013
 remote adjacency to GigabitEthernet0/2/0/0
 Prefix Len 32, traffic index 0, precedence routine (0), priority 3
  gateway array (0x1d42b878) reference count 3, flags 0x28000d0, source lsd (3), 0 backups
                [2 type 5 flags 0x10101 (0x1da3c230) ext 0x0 (0x0)]
  LW-LDI[type=5, refc=3, ptr=0x1d563344, sh-ldi=0x1da3c230]
   via 2.2.2.3, GigabitEthernet0/2/0/0, 0 dependencies, weight 0, class 0, protected [flags
 0x400]
```

```
    path-idx 0 bkup-idx 1 [0x1dc560cc 0x0]
    next hop 2.2.2.3
     local label 16011      labels imposed {16001}
  via 1.1.1.2, GigabitEthernet0/1/0/0, 2 dependencies, weight 0, class 0, backup (remote)
 [flags 0x300]
    path-idx 1 [0x1dc22110 0x0]
    next hop 1.1.1.2
    remote adjacency
     local label 16011      labels imposed {16002 16003}

    Load distribution: 0 (refcount 2)

    Hash  OK  Interface                 Address
    0     Y   GigabitEthernet0/2/0/0    remote

------------------------------------------------------------------------------------------------------

RP/0/RP0/CPU0:router# show mpls forwarding prefix 10.1.1.1/32

Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
16011  16001       10.1.1.1/32         SI0/2/CPU0   2.2.2.3         0
            { 16002 10.1.1.1/32         FI0/1/CPU0   1.1.1.2         0
               16003 }
------------------------------------------------------------------------------------------------------

RP/0/RP0/CPU0:router# show mpls forwarding prefix 10.1.1.1/32 detail

Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
16011  16001       10.1.1.1/32         SI0/2/CPU0   2.2.2.3         0
     Updated Apr 29 14:25:09.770
     Path Flags: 0x400 [ BKUP-IDX:1 (0x1dc460cc) ]
     Version: 5, Priority: 3
     MAC/Encaps: 0/4, MTU: 8000
     Label Stack (Top -> Bottom): { 16001 }
     Packets Switched: 0

       { 16002         10.1.1.1/32         FI0/1/CPU0   1.1.1.2         0
           16003 }
     Updated Apr 29 14:25:09.770
     Path Flags: 0x300 [ IDX:1 BKUP-REMOTE ]
     MAC/Encaps: 0/4, MTU: 1500
     Label Stack (Top -> Bottom): { 16002  16003 }
     Packets Switched: 0
------------------------------------------------------------------------------------------------------
RP/0/RP0/CPU0:router# show mpls forwarding prefix 10.1.1.1/32

Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
16011  16001       10.1.1.1/32         GI0/1/CPU0   2.2.2.3         0
            { 16002    10.1.1.1/32         GI0/2/CPU0   1.1.1.2         0
               16003 }
------------------------------------------------------------------------------------------------------


RP/0/RP0/CPU0:router# show mpls forwarding prefix 10.1.1.1/32 detail

Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
```

```
16011  16001       10.1.1.1/32         GI0/1/CPU0   2.2.2.3         0
     Updated Apr 29 14:25:09.770
     Path Flags: 0x400 [ BKUP-IDX:1 (0x1dc460cc) ]
     Version: 5, Priority: 3
     MAC/Encaps: 0/4, MTU: 8000
     Label Stack (Top -> Bottom): { 16001 }
     Packets Switched: 0

       { 16002         10.1.1.1/32         GI0/2/CPU0   1.1.1.2         0
           16003 }
     Updated Apr 29 14:25:09.770
     Path Flags: 0x300 [ IDX:1 BKUP-REMOTE ]
     MAC/Encaps: 0/4, MTU: 1500
     Label Stack (Top -> Bottom): { 16002  16003 }
     Packets Switched: 0
-------------------------------------------------------------------------------------------------
RP/0/RP0/CPU0:router# show cef fast-reroute

Prefix              Next Hop            Interface
10.1.1.1/32      2.18.6.2            Bundle-Ether100
                    13.1.11.2           GigabitEthernet0/6/1/9.11
10.2.1.1/32      2.18.6.2            Bundle-Ether100
                    13.1.11.2           GigabitEthernet0/6/1/9.11
10.3.1.1/32      2.18.6.2            Bundle-Ether100
                    13.1.11.2           GigabitEthernet0/6/1/9.11
-------------------------------------------------------------------------------------------------
```

MPLS LDP Verification

Verify the presence of remote backup paths in the MPLS LDP forwarding database.

```
RP/0/RP0/CPU0:router# show mpls ldp forwarding 10.1.1.1/32
Codes:
  - = GR label recovering, (!) = LFA FRR Pure Backup path
  {} = Label stack with multi-line output for a routing path
  G = GR, S = Stale, R = Remote LFA FRR Backup

Prefix           Label   Label          Outgoing     Next Hop            Flags
                 In      Out            Interface                        G S R
---------------- ------- -------------- ------------ ------------------- -----
10.1.1.1/32       16001   16002          Gi0/0/0/1   12.1.0.2
                          { 16003        Gi0/0/0/2    13.1.0.3        (!) G    R
                            16004 }                   (4.4.4.4)           G

-------------------------------------------------------------------------------

RP/0/RP0/CPU0:router# show mpls ldp forwarding 10.1.1.1/32 detail

Codes:
  - = GR label recovering, (!) = LFA FRR Pure Backup path
  {} = Label stack with multi-line output for a routing path
  G = GR, S = Stale, R = Remote LFA FRR Backup

Prefix           Label   Label          Outgoing     Next Hop            Flags
                 In      Out            Interface                        G S R
---------------- ------- -------------- ------------ ------------------- -----
10.1.1.1/32       16001   16002          Gi0/0/0/1    12.1.0.2
                         [ Protected; path-id 1 backup-path-id 33;
                           peer 2.2.2.2:0 ]
                         { 16003        Gi0/0/0/2    13.1.0.3        (!) G   R
                           16004 }                    (4.4.4.4)           G
                         [ Backup (remote); path-id 33; peer 3.3.3.3:0
                           remote LFA 4.4.4.4, remote peer 4.4.4.4:0 ]
```

```
  Routing update   : Apr 23 17:22:33.102 (00:08:02 ago)
  Forwarding update: Apr 23 17:22:47.183 (00:07:48 ago)
------------------------------------------------------------------------------


RP/0/RP0/CPU0:router# show mpls ldp forwarding summary

Forwarding Server (LSD):
  Connected: Yes
  Forwarding State Holdtime: 120 sec
Forwarding States:
…
  Rewrites:
Prefix:
      Total: 5 (4 with ECMP, 1 LFA FRR protected)
      Labelled:
…
    Paths:
      Total: 10 (1 backup [1 remote], 2 LFA FRR protected)
      Labelled: 10 (1 backup)
```

MPLS LSD Verification

Verify the presence of remote backup paths in the MPLS LSD forwarding database.

```
RP/0/RP0/CPU0:router# show mpls lsd forwarding labels 16001

In_Label, (ID), Path_Info: <Type>
16001, (IPv4, 'default':4U, 10.1.1.1/32), 2 Paths
   1/2: IPv4-rLFA, 'default':4U, Gi0/0/0/1, nh=12.1.0.2, lbl=16002, tun_id=0,
            flags=0x0 () [Protected]
   2/2: IPv4-rLFA, 'default':4U, Gi0/0/0/2, nh=13.1.0.3, lbls={ 16003, 16004 }, tun_id=0,
    flags={ 0x4008 (Retain, Remote-Backup), 0x8 (Retain) } [Backup]
-------------------------------------------------------------------------------------------

RP/0/RP0/CPU0:router# show mpls lsd forwarding labels 16001 detail

In_Label, (ID), Path_Info: <Type>
16001, (IPv4, 'default':4U, 10.1.1.1/32), 2 Paths
   1/2: IPv4-rLFA, 'default':4U, Gi0/0/0/1, nh=12.1.0.2, lbl=16002, tun_id=0,
            flags=0x0 () [Protected]
            path-id=1, backup-path-id=33
   2/2: IPv4-rLFA, 'default':4U, Gi0/0/0/2, nh=13.1.0.3, lbls={ 16003, 16004 }, tun_id=0,
            flags={ 0x4008 (Retain, Remote-Backup), 0x8 (Retain) } [Backup]
            path-id=33, backup-path-id=0
BCDL priority:3, LSD queue:15, version:103
      Installed Apr 23 17:22:47.183 (00:17:09 ago)
-------------------------------------------------------------------------------------------

RP/0/RP0/CPU0:router# show mpls lsd forwarding summary

Messages: 22
Forwarding updates: 34
Rewrites: 9
  FPIs:
    Label: 9
    IPv4: 5
..
  MOIs: 13
    IPv4 paths: 10 (1 backup [1 remote], 2 protected)
..
    IP subscriber: 0
```

Ingress Forwarding Chain Verification (Backup over Non-TE)

Verify the presence of remote backup paths in the ingress forwarding chain.

```
RP/0/RP0/CPU0:router#show cef 10.1.1.1 hard in det loc 0/1/CPU0
Sat May 12 05:16:47.929 UTC
10.10.10.10/32, version 5475, internal 0x4004001 (ptr 0x5f9582f0) [1], 0x0 (0x5f29cf94),
0x450 (0x6058e35c)
 Updated May 12 05:05:51.294
...

 Print Flags: 00000000

  INGRESS PLU
   SW: 08300200 00000004 00040000 00263b00
   HW: 08300200 00000004 00040000 00263b00
  entry_type:         FWD    rpf ptr:     0x00300200
  prefix len:           1    BGP policy a/c:       0
QoS group:            0    BAO id:               0
  num entries:          1    next ptr:    0x0000263b
  label ptr:   0x00000000    Label(0) Ptr(0)

    Load info:
    Flag: 0x00860001
    TLU Channel: 1 Addr: 0x0000263b
    ENTRY        0
      SW: 00000000 00712426 b4040002 00000001
      HW: 00000000 00712426 b4040002 00000001
    PBTS:              0    extra l3li:             0
    entry type:      FWD    next ptr:     0x00712426
    is label:    0    is label ptr:    0
    num of entries:    1
    tunnel_encap_ptr: 0x00000000
    next-hop:   180.4.0.2

       TLU Channel: 2 Addr: 0x00712426
       ENTRY        0
        SW: 00000008 00000000 03e82000 20460b00
        HW: 00000008 00000000 03e82000 20460b00
       label1:        16002    label2:            16003
        label 3: 16009
       num of labels:     1    next ptr: 0x0020460b


  frr Flags        : 0x28c /* New flag to indicate that LFA is over PQ */
 Primary adjacency
 L2 Load info
  TLU Channel: 3 Addr: 0x20460b
  [HW: 0x00010000 0x00000000 0x00000000 0x30040b00]
    num. entries  : 1
    num. labels   : 0
    label 1       : 0
next ptr      : 0x30040b
 L2 Load Balancing Entry
  TLU Channel: 4 Addr: 0x30040b
    Entry[0]
    [HW: 0x00000004 0x00000084 0x01280440 0x00050000]
      dest. addr         : 0x4
      sponge queue       : 132
      egress port        : 0x128044
      rp destined        : no
      service destined   : no
      rp drop            : no
      hash type          : 0
```

```
     uidb index         : 0x5

FRR backup info
 FRR Flags                  : 0x28c  /* updated to indicate PQ is active */
Cached backup adjacency
 Cached backup num. entries : 1
     protected num. entries : 1
     backup is a tunnel     : yes
     backup tunnel local label : 16007
     remote LFA active: 1
 Shared TLU Channel: 4 Addr: 0x300a02
   Entry[0]
   [HW: 0x00000004 0x00000086 0x01280480 0x00070000]
     dest. addr         : 0x4
     sponge queue       : 134
     egress port        : 0x128048
     rp destined        : no
     service destined   : no
     rp drop            : no
     hash type          : 0
     uidb index         : 0x7

   Load distribution: 0 (refcount 3)

   Hash  OK  Interface                Address
   0     Y   TenGigE0/2/0/4           180.4.0.2
```

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 Cisco NCS 6000 Series 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 309.

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
!
```

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* module.

To build an MPLS TE topology, create tunnels, and configure forwarding over the tunnel for OSPF Version 2; see *MPLS Configuration Guide for Cisco NCS 6000 Series 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	*Routing Command Reference for Cisco NCS 6000 Series Routers*
MPLS TE feature information	*Implementing MPLS Traffic Engineering on* module in *MPLS Configuration Guide for Cisco NCS 6000 Series Routers*

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

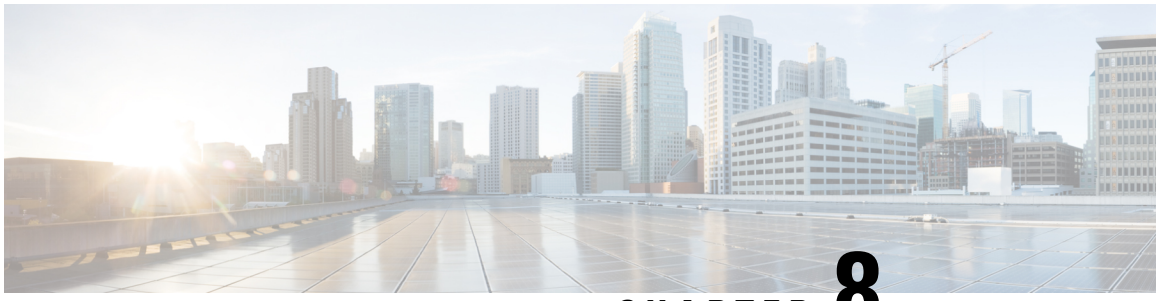
RFCs

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)
RFC 4124	Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering
RFC 4750	OSPF Version 2 Management Information Base
RFC 4811	OSPF Out-of-Band Link State Database (LSDB) Resynchronization

RFCs	Title
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 393
- Restrictions for Loop-Free Alternate Fast Reroute, on page 393
- IS-IS and IP FRR, on page 394
- Repair Paths, on page 394
- LFA Overview, on page 395
- LFA Calculation, on page 395
- Interaction Between RIB and Routing Protocols, on page 395
- Configuring Fast Reroute Support, on page 396
- Configuring IPv4 Loop-Free Alternate Fast Reroute Support: Example, on page 398
- Additional References, on page 398

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** *process-id* **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. • 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
		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. • 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	**net** *net* **Example:** `RP/0/RP0/CPU0:router(config-isis)# net 47.0001.0000.0000.8888.00`	Configures an IS-IS network entity (NET) for a routing process.
Step 5	**address-family** {**ipv4** | **ipv6**} [**unicast** | **multicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters the interface address family configuration mode.
Step 6	**metric-style wide** **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide`	Configures a router to generate and accept wide link metrics only.
Step 7	**exit** **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# exit`	Exits router address family configuration mode, and resets the router to router configuration mode.
Step 8	**interface** *bundle bundle-id* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface Bundle-Ether 9`	Creates and names a new Ethernet link bundle.
Step 9	**address-family** {**ipv4** | **ipv6**} [**unicast** | **multicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters interface address family configuration mode.
Step 10	**fast-reroute per-prefix** **Example:**	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

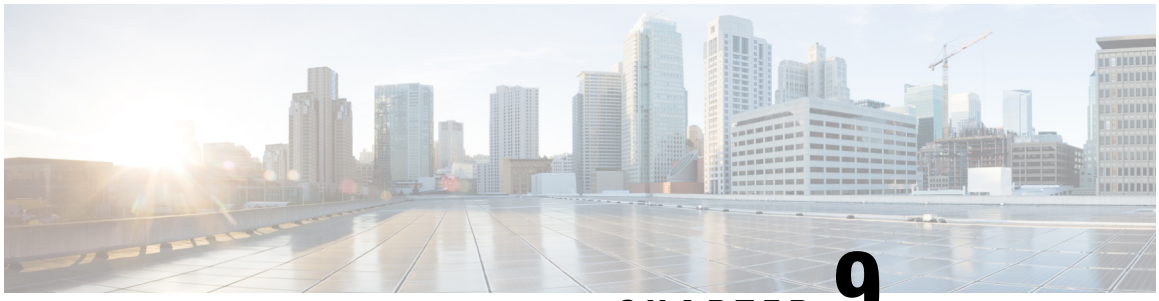
Related Topic	Document Title
IS-IS commands	*Routing Command Reference for Cisco NCS 6000 Series Routers*
MPLS commands	*Routing Command Reference for Cisco NCS 6000 Series Routers*

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 .

Feature History for Implementing and Monitoring RIB

Release 5.0.0	This feature was introduced.

- Prerequisites for Implementing RIB, on page 401
- Information About RIB Configuration, on page 402
- How to Deploy and Monitor RIB, on page 405
- Configuring RCC and LCC, on page 407
- Configuration Examples for RIB Monitoring, on page 408
- Where to Go Next, on page 411
- Additional References, on page 411

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. 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). RIB processes are not responsible for the BRIB, which are handled by BGP.

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 7: 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

In Cisco IOS XR software, RIB tables support unicast routing.

The default routing tables for Cisco IOS XR software RIB are the unicast RIB tables for IPv4 routing.

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.

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 NCS 6000 Series 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 XR 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.

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 ipv4** | **ipv6**] [**unicast**]
2. **show route ipv4** | **ipv6**] [**unicast**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show route ipv4** | **ipv6**] [**unicast**] **Example:** `RP/0/RP0/CPU0:router# show route summary`	Displays route summary information about the specified routing table. • The default table summarized is the IPv4 unicast routing table.
Step 2	**show route ipv4** | **ipv6**] [**unicast**] **Example:** `RP/0/RP0/CPU0:router# show route ipv4 unicast`	Displays more detailed route information about the specified routing table. • 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.

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

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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring RCC and LCC

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

1. **configure**
2. Use one of these commands:
   - **rcc** {**ipv4** | **ipv6**} **unicast** {**enable** | **period** *milliseconds*}
   - **lcc** {**ipv4** | **ipv6**} **unicast** {**enable** | **period** *milliseconds*}
3. Use the **commit** or **end** command.
4. 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** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	Use one of these commands: • **rcc** {**ipv4**	**ipv6**} **unicast** {**enable**	**period** *milliseconds*} • **lcc** {**ipv4**	**ipv6**} **unicast** {**enable**	**period** *milliseconds*} **Example:** `RP/0/RP0/CPU0:router(config)#rcc ipv6 unicast enable`	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).

	Command or Action	Purpose

	`RP/0/RP0/CPU0:router(config)#rcc ipv6 unicast period 500`
	Or
	`RP/0/RP0/CPU0:router(config)#lcc ipv6 unicast enable`
	`RP/0/RP0/CPU0:router(config)#lcc ipv6 unicast period 500`	

Step 3	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 4	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*]
	Example:
	`RP/0/RP0/CPU0:router#show rcc ipv6 unicast statistics scan-id 120`
	Or
	`RP/0/RP0/CPU0:router#show lcc ipv6 unicast statistics scan-id 120`
		Displays statistics about background scans.
		• **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
```

Where to Go Next

For additional information on the protocols that interact with RIB, you may want to see the following publications:

- *Implementing BGP* in *Routing Configuration Guide for Cisco NCS 6000 Series Routers*
- *Implementing EIGRP* in *Routing Configuration Guide for Cisco NCS 6000 Series Routers*
- *Implementing IS-IS* in *Routing Configuration Guide for Cisco NCS 6000 Series Routers*
- *Implementing OSPF* in *Routing Configuration Guide for Cisco NCS 6000 Series Routers*
- *Implementing RIP* in *Routing Configuration Guide for Cisco NCS 6000 Series Routers*
- *RIB Commands* in *Routing Command Reference for Cisco NCS 6000 Series 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	*RIB Commands on Cisco IOS XR Software* in *Routing Command Reference for Cisco NCS 6000 Series Routers*

Standards and RFCs

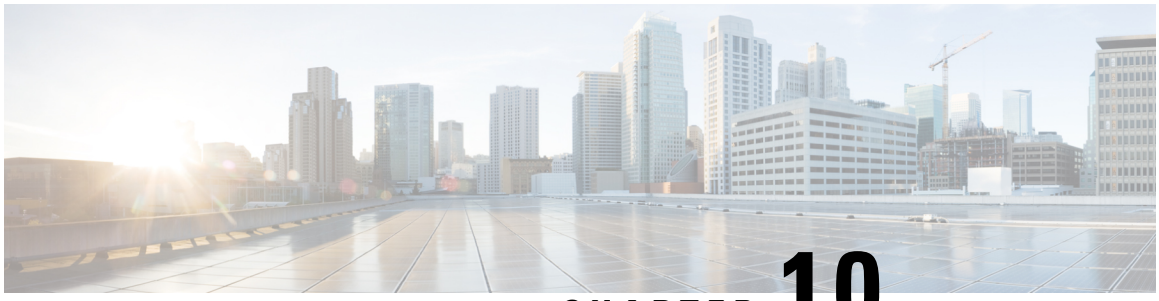
Standard/RFC	Title
Draft-ietf-rtgwg-ipfrr-framework-06.txt	*IP Fast Reroute Framework*, by M. Shand and S. Bryant
Draft-ietf-rtgwg-lf-conv-frmwk-00.txt	*A Framework for Loop-free Convergence*, by M. Shand and S. Bryant
No new or modified RFCs are supported by this feature, and support for existing RFCs 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
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/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.

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 429 section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the .

Feature History for Implementing RIP

Release 5.0.0	This feature was introduced.

- Prerequisites for Implementing RIP, on page 413
- Information About Implementing RIP, on page 414
- How to Implement RIP, on page 419
- Configuration Examples for Implementing RIP, on page 427
- Additional References, on page 428

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

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 421 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 8: 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
OSPF	110
IS-IS	115
RIP version 1 and 2	120
External EIGRP	170

Routing Protocols	Administrative Distance Value
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 NCS 6000 Series Routers*
- *Keychain Management Commands* module in *System Security Command Reference for Cisco NCS 6000 Series 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, 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.
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. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters XR Config mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# configure`	
Step 2	**router rip** **Example:** `RP/0/RP0/CPU0:router(config)# router rip`	Configures a RIP routing process.
Step 3	**neighbor** *ip-address* **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	**broadcast-for-v2** **Example:** `RP/0/RP0/CPU0:router(config-rip)# broadcast-for-v2`	(Optional) Configures RIP to send only Version 2 packets to the broadcast IP address. This command can be applied at the interface or XR Config level.
Step 5	**interface** *type interface-path-id* **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 6	**receive version** { **1** | **2** | **1 2** } **Example:** `RP/0/RP0/CPU0:router(config-rip-if)# receive version 1 2`	(Optional) Configures an interface to accept packets that are: • Only RIP v1 • Only RIP v2 • Both RIP v1 and RIP v2
Step 7	**send version** { **1** | **2** | **1 2** } **Example:** `RP/0/RP0/CPU0:router(config-rip-if)# send version 1 2`	(Optional) Configures an interface to send packets that are: • Only RIP v1 • Only RIP v2 • Both RIP v1 and RIP v2
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
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** *update invalid holddown flush*	(Optional) Adjusts RIP network timers.

	Command or Action	Purpose
	Example: `RP/0/RP0/CPU0:router(config-rip)# timers basic 5 15 15 30`	**Note** To view the current and default timer values, view output from the **show rip** command.
Step 5	**output-delay** *delay* **Example:** `RP/0/RP0/CPU0:router(config-rip)# output-delay 10`	(Optional) Changes the interpacket delay for the RIP updates sent. **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:** `RP/0/RP0/CPU0:router(config-rip)# nsf`	(Optional) Configures NSF on RIP routes after a RIP process shutdown or restart.
Step 7	**interface** *type interface-path-id* **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 8	**metric-zero-accept** **Example:** `RP/0/RP0/CPU0:router(config-rip-if)# metro-zero-accept`	(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:** `RP/0/RP0/CPU0:router(config-rip-if)# split-horizon disable`	(Optional) Disables the split horizon mechanism. • 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.
Step 10	**poison-reverse** **Example:** `RP/0/RP0/CPU0:router(config-rip-if)# poison-reverse`	Enables poison reverse processing of RIP router updates.
Step 11	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.

	Command or Action	Purpose
Step 2	**router rip** **Example:** `RP/0/RP0/CPU0:router(config)# router rip`	Configures a RIP routing process.
Step 3	**neighbor** *ip-address* **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** *type interface-path-id* **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:** `RP/0/RP0/CPU0:router(config-rip-if)# passive-interface`	(Optional) Suppresses the sending of RIP updates on an interface, but not to explicitly configured neighbors.
Step 6	exit **Example:** `RP/0/ /CPU0:router(config-rip-if)# exit`	(Optional) Returns the router to the next higher configuration mode.
Step 7	**interface** *type interface-path-id* **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.

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. Use the **commit** or **end** command.
6. **configure**
7. **router rip**
8. **route-policy** *route-policy-name* { **in** | **out** }
9. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**route-policy** *name* **Example:** `RP/0/RP0/CPU0:router(config)# route-policy IN-IPv4`	Defines a route policy and enters route-policy configuration mode.
Step 3	**set rip-metric** *number* **Example:**	(Optional) Sets the RIP metric attribute.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-rpl)# set rip metric 42`	
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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 6	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 7	**router rip** **Example:** `RP/0/RP0/CPU0:router(config)# router rip`	Configures a RIP routing process.
Step 8	**route-policy** *route-policy-name* { **in** | **out** } **Example:** `RP/0/RP0/CPU0:router(config-rip)# route-policy rp1 in`	Applies a routing policy to updates advertised to or received from an RIP neighbor.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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 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
!
```

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	*Routing Command Reference for Cisco NCS 6000 Series Routers*
Site of Origin (SoO) support for RIP feature information	*Implementing MPLS Traffic Engineering on* module in the *MPLS Configuration Guide for Cisco NCS 6000 Series Routers*
Cisco IOS XR getting started documentation	
Information about user groups and task IDs	*Configuring AAA Services on* module in the *System Security Configuration Guide for Cisco NCS 6000 Series Routers*

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

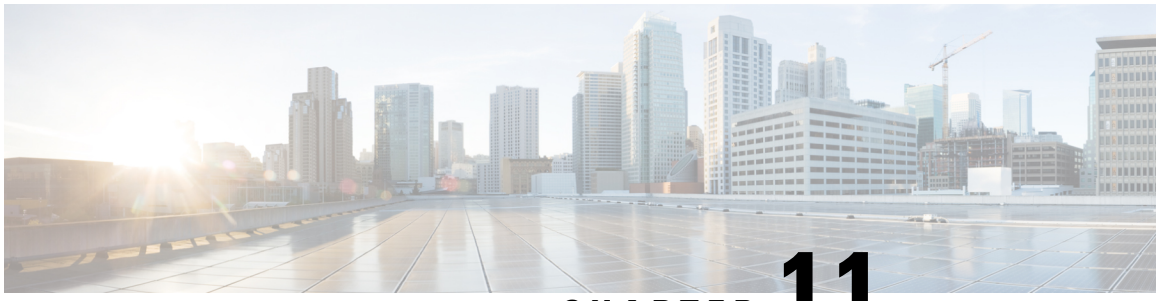
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

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 515 section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the .

Feature History for Implementing Routing Policy

Release 5.0.0	This feature was introduced.

- Prerequisites for Implementing Routing Policy, on page 431
- Restrictions for Implementing Routing Policy, on page 432
- Information About Implementing Routing Policy, on page 433
- How to Implement Routing Policy, on page 504
- Configuration Examples for Implementing Routing Policy, on page 508
- Additional References, on page 515

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.
- You cannot change the next hop address to an IPv6 address through RPL policy for a route that starts from an IPv4 peer.
- 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**.

- Default number of lines of config (both policies and sets) to be configured in the system is 65536 (64000).
- Default number of policies to be configured in the system is 3500.
- Maximum number of policies to be configured in the system is 5000.
- Maximum number of lines of config (both policies and sets) to be configured in the system is 131072 (128000).
- Maximum number of conditions in a policy statement *[if conditions]* to be configured in the system is 16.
- Maximum depth of policy statements *[if depth]* to be configured in the system is 64.
- Maximum length of policy name is 66.
- Irrespective of the number of elements in a set, the length of a set as 3 is considered.

The **show rpl maximum** comamnd on the router or the device shows the number of policies configured, current limit and max limit.

```
Router#show rpl maximum
         Tue Aug 30 16:18:53.497 IST
```

```
                          Current    Current       Max
                           Total      Limit       Limit
         ------------------------------------------------------------
        Lines of configuration            6      65536     131072
        Policies                          1       3500       5000
        Compiled policies size (kB)       0
```

You can modify policy limit using the following CLI:

```
Router(config)#rpl maximum policies ?
         <1-5000>  Enter the number of policies limit.
```

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.

User can use the command **show rpl regexp** to evaluate the performanceofios-regex and dfa-regex engines, which will display the execution time of both the engines.

Examples:

```
Router#show rpl regexp
'_(174|209|286|701|1239|1299|2828|2914|3257|3320|3356|3549|5511|6453|6461|6762|7018|12956)_'
 '6461 6461 6461 6461 4637 4637 4637 4637 1221$'

Regular Expression
_(174|209|286|701|1239|1299|2828|2914|3257|3320|3356|3549|5511|6453|6461|6762|7018|12956)_
Input string 6461 6461 6461 6461 4637 4637 4637 4637 1221$

Regex Engine            Match             Execution Time
------------------------------------------------------------
ios-regex               PASS              7us
dfa-regex               PASS              37us
```

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 435, community-set, on page 436, extcommunity-set, on page 437, prefix-set, on page 439, and rd-set, on page 441. 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 445 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.

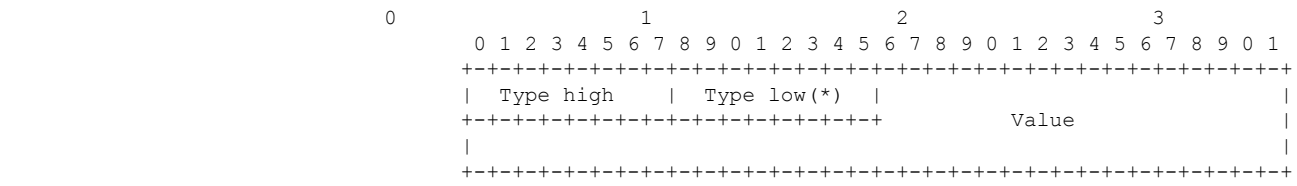
extcommunity-set

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. 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 extended communities attribute is a transitive optional BGP attribute, with the Type Code 16. Each extended community is encoded as an 8-octet quantity, as follows: - Type Field: 1 or 2 octets - Value Field: Remaining octets



For more information, see RFC4360.

The following types of extended community sets are supported:

- bandwidth—Advertises the bandwidth of an autonomous system exit link as an extended community for links between directly connected external BGP (eBGP) neighbors. The extended community is used with BGP multipath features to configure load balancing over links with unequal bandwidth. When this community set is enabled, the routes learned from the external neighbors are propagated through the internal BGP (iBGP) network with the bandwidth of the source external link.

  ```
  extcommunity-set bandwidth extcomm-bw
    100:25000
  end-set
  ```

  The demilitarized zone (DMZ) link-bandwidth value is configured using outbound route-policy using routing table or adding the *additive* keyword. If *additive* keyword is not added, other extended communities such as route targets are removed. Removal of route targets will lead to the routes-not-imported condition at the receiving end of the peer.

  ```
  extcommunity-set bandwidth dmz_ext
     1:8000
   end-set
  ```

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

Note In the above example,

- *1:8000*—The first number (1) indicates the ASN of the router sending the extended community. This ASN is a 16-bit decimal number. The second number (8000) indicates the bandwidth in bytes per second. The bandwidth is a 32-bit decimal number.

- cost—Allows to customize the BGP best path selection process for a local autonomous system or confederation. The cost community is a nontransitive, extended community attribute that is passed to iBGP and confederation peers, but not to eBGP peers. The cost community attribute is applied to internal routes in a route map. The cost community set clause is configured with a cost community ID number (0-255). The cost community ID number determines the preference for the path selection process. The path with the lowest cost community ID number is preferred.

```
extcommunity-set cost a_cost_set
    IGP:1:10
  end-set
```

- opaque—Configures the color extended community.

```
extcommunity-set opaque a_opaque_set
  12345
end-set
```

- rt—Identifies a set of sites and VRFs that may receive routes tagged with the configured route target. Configuring the route target (RT) extended community attribute with a route allows that route to be placed in the per-site forwarding tables to route traffic that is received from corresponding sites. The attribute adds the route target extended community attributes to the VRF's list of import, export, or both (import and export) route-target extended communities. You can also specify route targets in the import and export statements under Global VRF configuration.

```
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)
  ($ipadrr:666, 1234:$tag, 1.2.3.4:777, $tag2:777)
```

- seg-nh—Indicates the Point-to-Multipoint (P2MP) segmented next-hop extended community to indicate LSPs should be segmented when the BGP MVPN I-PMSI or S-PMSI autodiscovery (AD) routes are advertised or propagated to signal inter-area P2MP service. This extended community must be included in the I-PMSI or S-PMSI AD route by the PE that originates such a route, or an ASBR that re-advertises such a route into its own AS.

```
extcommunity-set seg-nh seg_set
  1.1.1.1
end-set
```

- soo—The site-of-origin (SoO) extended community is a BGP extended community attribute that is used to identify routes that have originated from a site so that the readvertisement of that prefix back to the source site can be prevented. The SoO extended community uniquely identifies the site from which a router has learned a route. BGP can use the SoO value associated with a route to prevent routing loops between CEs in the same site when AS-override is used .

  The extended community value takes one of the following formats:

  - 16-bit autonomous system number, a colon, and a 32-bit number
  - 32-bit IP address, a colon, and a 16-bit number

```
extcommunity-set soo a_soo_set
  1.1.1.2:51,
  100:200
end-set
```

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 **g**reater than or **e**qual 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 **l**ess than or **e**qual 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 **eq**ual 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>)

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 504 for detailed information on how to implement routing policy language.

Pass PolicyPass 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 445, 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 gbpathtype 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
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 NCS 6000 Series 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 are dampened than the routes that are less specific than /25.

Note When the dampening policy runs for a route, then the last "set dampening" statement that is encountered, takes effect.

- If a "drop" statement is encountered, then the route is not dampened; even if the "set dampening" statement is encountered.
- If a "pass" or "done" statement is encountered but not the "set dampening" statement, then the route is dampened using the default dampening parameters.

For example:

- When policy1 applies another policy that is called policy2 and if a "pass" statement is encountered in policy2, then policy2 exits and continues to execute policy1.
- If a "done" statement is encountered in policy2, then both policy1 and policy2 exits immediately.

```
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-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
  .
  .
  .
```

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 NCS 6000 Series 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 NCS 6000 Series 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
```

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 bgpvrf-export
    .
    .
    .
```

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.

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 15.15.15.15.15:15. 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
    .
    .
    .
```

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 9: 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 458)
	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	n/a
	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	n/a
	next-hop	in	set set peer address
	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	
	weight	n/a	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		

Attach Point	Attribute	Match	Set	
			set set+ set-	
next-hop		in	set set self	
origin		is	set	
path-type		is	n/a	
rd		in	n/a	
source		in	n/a	
unsuppress-route		n/a	unsuppress-route	
weight		is	n/a	
neighbor-orf		orf-prefix	in	n/a

Attach Point	Attribute	Match	Set
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 10: OSPF 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
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

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-redist. 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-redist
  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-redist
  .
  .
  .
```

OSPFv3 Attributes and Operators

This table summarizes the OSPFv3 attributes and operators per attach points.

Table 11: 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 12: 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: *ospf-nssa-type-1* and *ospf-nssa-type-2*	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 13: 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 redist-ospf
  set rip-metric 5
end-policy

router rip
  redistribute ospf 1 route-policy redist-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
```

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

In RPL, you can use the **rpl editor vim** configuration to update the default editor.

```
Router(config)#rpl editor ?
  emacs  Set default RPL editor to Emacs
  nano   Set default RPL editor to nano
  vim    Set default RPL editor to 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 XR 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 XR 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 XR Config 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 polices 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 499.

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)# 198.51.100.1/32
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.

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.

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.

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)# 192.0.2.1:777,
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.

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.

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 are 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. In other words, if the import RT is already defined, it will still add the RTs mentioned in the policy to the imported route-targets list but without the use of the **import** command.

Use the **source rt import-policy** command under VRF sub-mode of VPN address-family configuration mode to enable this feature.

Configuring VRF Import Policy

```
/* Configure import policy */
/* The below task configures import policy. However, it does not enable importing of routes.
 */
Router(config)# route-policy VRF_Import
Router(config-rpl)# if extcommunity rt matches-any (65500:1000) and destination in
```

```
(10.28.0.128/28) then
Router(config-rpl-if)# pass
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy

/* Enable the import of routes */
The below task enables the import of routes. */
Router(config)# vrf vrf1
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# import route-policy VRF_Import
Router(config-vrf-af)# export route-target 65500:2000

/* Enable the import of routes using the source rt command */
/* The below task enables the route-targets to be imported from the import-policy.
There is no need to explicit configure the import command. If you configure the vrf vrf1
command, routes with RT 65500:1000 are imported. If you configure the import command, that
 only adds to the list of route-targets to import. */

Router(config)# router bgp 1
Router(config-bgp)# address-family vpnv4 unicast
Router(config-bgp-af)# vrf all
Router(config-bgp-af-vrf-all) source rt import-policy
```

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.

Tip You can programmatically configure the route policy using `openconfig-routing-policy.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco NCS 6000 Series Routers*.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name* [*parameter1* , *parameter2* , . . . , *parameterN*]
3. **end-policy**

4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**route-policy** *name* [*parameter1* , *parameter2* , . . . , *parameterN*] **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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** | **ipv6** } **unicast**

5. **route-policy** *policy-name* { **in** | **out** }
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 125`	Configures a BGP routing process and enters router configuration mode. • The *as-number* 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.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.20`	Specifies a neighbor IP address.
Step 4	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`	Specifies the address family.
Step 5	**route-policy** *policy-name* { **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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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" section 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** } *name* [**nano** | **emacs** | **vim** | **inline** { **add** | **prepend** | **remove** } *set-element*] **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. • 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. • 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** [*name* [**detail**] | **states** | **brief**] **Example:** `RP/0/RP0/CPU0:router# show rpl route-policy sample2`	(Optional) Displays the configuration of a specific named route policy. • 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** [*name* | **states** | **brief**] **Example:**	(Optional) Displays the contents of a named prefix set. • To display the contents of a named AS path set, community set, or extended community set, replace

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# show rpl prefix-set prefixset1`	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)# 198.51.100.1/32
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.

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.

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.

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)# 192.0.2.1:777,
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.

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.

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	*Routing Policy Language Commands on* module of the *Routing Command Reference for Cisco NCS 6000 Series Routers*
Regular expression syntax	*Understanding Regular Expressions, Special Characters and Patterns* appendix in the

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

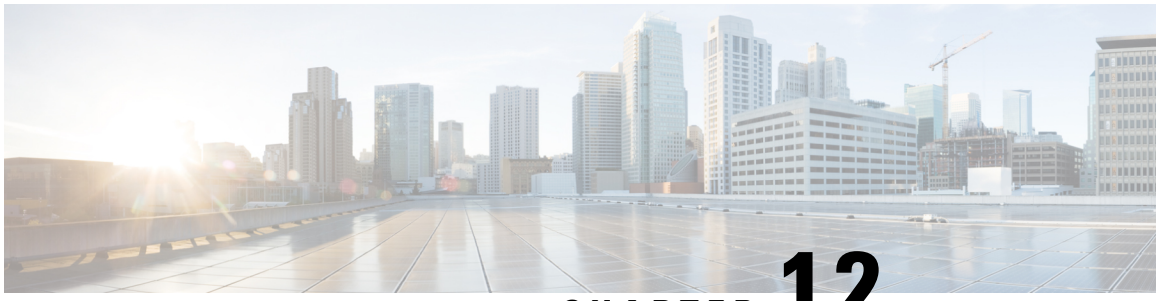
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

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 522 section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the .

Feature History for Implementing Static Routes

Release 5.0.0	This feature was introduced.

- Prerequisites for Implementing Static Routes, on page 517
- Restrictions for Implementing Static Routes, on page 517
- Information About Implementing Static Routes, on page 518
- Configuration Examples, on page 520
- Where to Go Next, on page 521
- Additional References, on page 521

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. 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.

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
```

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 NCS 6000 Series Routers*
- *Implementing BGP on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco NCS 6000 Series Routers*
- *Implementing EIGRP on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco NCS 6000 Series Routers*
- *Implementing IS-IS on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco NCS 6000 Series Routers*
- *Implementing OSPF on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco NCS 6000 Series Routers*
- *Implementing OSPFv3 on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco NCS 6000 Series Routers*
- *RIB Commands on Cisco IOS XR Software* in *Routing Command Reference for Cisco NCS 6000 Series Routers*
- *Implementing RIP on Cisco IOS XR Software* in *Routing Configuration Guide for Cisco NCS 6000 Series 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	*Static Routing Commands* in *Routing Command Reference for Cisco NCS 6000 Series Routers*

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.	NA

MIBs

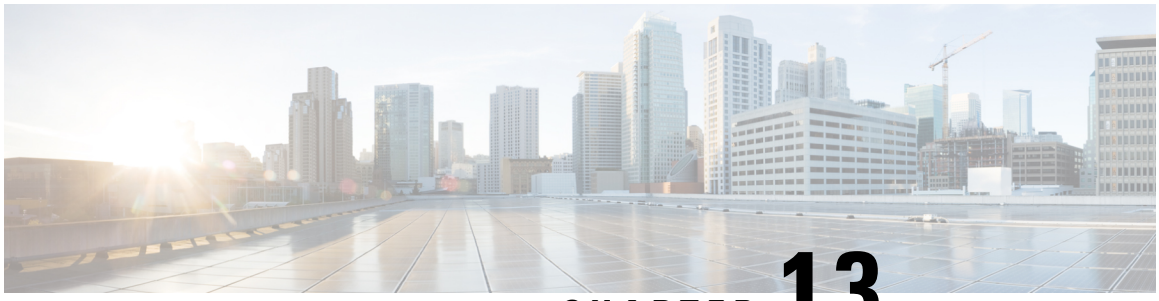
MIBs	MIBs Link
All	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator tool found at the following URL and choose a platform under the Cisco Access Products menu.

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.	NA

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 5.0.0	This feature was introduced.

- Route Convergence Monitoring and Diagnostics, on page 523
- Configuring Route Convergence Monitoring and Diagnostics, on page 524

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:

- 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.

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** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
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** *location* **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** *number* **Example:**	Sets event buffer size 9as number of events) for storing event traces .

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-rcmd)#event-buffer-size 100`	
Step 5	**max-events-stored** *number* **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** *minutes* **Example:** `RP/0/RP0/CPU0:router(config-rcmd)#monitoring-interval 120`	Sets interval (in minutes) to collect logs.
Step 7	**node** *node-name*	Configures parameters for a specified node. `RP/0/RP0/CPU0:router(config-rcmd)#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.
		• ISIS-Select ISIS to configure parameters related to ISIS protocol
		• OSPF-Select OSPF to configure parameters related 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.
		• Critical-Set to monitor route convergence for critical priority routes
		• High-Set to monitor route convergence for high priority routes
		• 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-proto-prio)#disable`	Disables the monitoring of route convergence for specified priority.
Step 11	**leaf-network** *number* **Example:** `RP/0/RP0/CPU0:router(config-rcmd-proto-prio)#leaf-network 100`	Enables leaf network monitoring. Specify a maximum number of leaf networks to be monitored. Range for maximum number is 10-100.

	Command or Action	Purpose
Step 12	**threshold** *value* **Example:** `RP/0/RP0/CPU0:router(config-rcmd-proto-prio)#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-store)#`
		Sets the absolute directory path for storing diagnostic reports.

Step 14	**diagnostics** *directory-path-name* **Example:** `RP/0/RP0/CPU0:router(config-rcmd-store)#diagnostics /disk0:/rcmd`	Specifies the absolute directory path for storing diagnostic reports. Set a directory-path-name. Example: /disk0:/rcmd/ or <tftp-location>/rcmd/
Step 15	**diagnostics-size** **Example:** `RP/0/RP0/CPU0:router(config-rcmd-store)# diagnostics-size 8`	Specify a maximum size for the diagnostics directory. Set the size in %. Range is 5%-80%.
Step 16	**reports** *directory-path-name* **Example:** `RP/0/RP0/CPU0:router(config-rcmd-store)#reports /disck0:/rcmd`	Specifies the absolute directory path for storing reports. Set a directory-path-name. Example: /disk0:/rcmd/ or <tftp-location>/rcmd/
Step 17	**reports-size** **Example:** `RP/0/RP0/CPU0:router(config-rcmd-store)#reports-size 8`	Specify a maximum size for the reports directory. Set the size in %. Range is 5%-80%.

INDEX

A

ABRs (area border routers) 304
ABRs (area border routers) 304
action 454
additional-path attach point 457
address family command 12, 235
adjacencies, tuning 265
adjacency 316
adjacency (OSPFv2) 307
administrative distance 402, 416, 518
administrative distance, static routes 518
allocate label 464
apply command 456
Area Border Routers (ABRs) 304
area command (BFD) 166–169
area-in 481
area-out 482
as-path-set, inline set form 435
as-path-set, named set form 435
ASBRs (autonomous system boundary routers) 305
attached 492
attached bit on an instance 242
attached bit on an IS-IS instance 242
attaching to BGP neighbor 505
attributes 445, 447
authentication 306–307, 330
  configuring (OSPFv2) 330
  MD5 (OSPFv2) 306
  route, key rollover (OSPFv2) 307
  strategies 306
authentication using keychain 219
authentication using keychain in RIP 417
authentication, configuring 260, 330
autonomous system number format 10
autonomous systems 303

B

backbone area 304
bandwidth 423
benefits 414
bestpath algorithm 28
BFD 147, 149–150, 164–169, 177–179, 182–187
  BFD configuration mode 177–179, 182–186
BFD *(continued)*
  BGP configuration mode 164–165
  counters 187
    clearing 187
    displaying 187
  dampening, configuring 184
  echo mode, disabling 182–183
  echo mode, specifying source address 177–178, 182
  enabling 164–166
    interface 166
    local device and peer, between 165
    neighbor 164
  fast detection, configuring 166–169
  ipv6 checksum, enabling or disabling 185–187
  latency detection, configuring 179
  OSPF 166–167
    configuration mode 166–167
  OSPFv3 configuration mode 168–169
  overview 149
  prerequisites 147
  setting 165–169
    BFD multiplier 165
    minimum interval 165–169
    multiplier 166–169
  source and destination ports 150
bfd command 177–179, 182–186
bfd fast-detect command 165–169
bfd minimum-interval command 165–169
BFD multihop support for BGP 50
bfd multiplier command 165–169
BFD on BGP 196
  example 196
BFD on OSPF 197
  example 197
distance bgp command 79–80
weight command 69
keychain command 97
shutdown command 98
timers bgp command 65–66
BGP (Border Gateway Protocol) 3–5, 9–14, 18, 23–24, 28, 40, 111, 458–466
  autonomous system number format 10
  bestpath algorithm 28
  BGP keychains 40

BGP (Border Gateway Protocol) *(continued)*
bgp router submode 11
router bgp command 11
bidirectional forwarding detection 4
configuration 12, 14
grouping 12
inheriting 14
inheriting templates 14
default address family 40
description 3
functional overview 4
inheritance, monitoring 18
IPv6 prefixes over IPv4 session 9
neighbors, maximum limits on 5
policy attach points 458–466
allocate label 464
clear policy 465
dampening 458
debug 466
default originate 458
export 463
import 462
neighbor export 459
neighbor import 459
neighbor-orf 464
network 460
next-hop 465
retain route target 464
show bgp 461
router bgp neighbor group address family configuration mode,address family command 13
router identifier 4
routing policy, enforcing 23
update groups 24, 111
description 24
example 111
bgp add path 46
BGP configuration 25
bgp cost community 25
bgp dampening command 77
table-policy command 78–79
aggregate-address command 73
bgp global address family submode 12
address family command 12
BGP keychains 40
bgp multi-as 50
bgp multi-instance 50
next-hop-self command 89
route-policy (BGP) command 85–86
route-reflector-client command 83–84
soft-reconfiguration inbound command 95–96
bgp neighbor address family submode 12, 235
neighbor address family command 12
bgp neighbor command 12
bgp neighbor group submode 13
neighbor-group command 13
bgp neighbor submode 12, 235
bgp neighbor command 12
neighbor command 12, 235
BGP Non-Stop Routing 105
configuring 105
BGP nonstop routing 43
bgp nsr configuration example 114
bgp bestpath as-path ignore command 70–71
bgp default local-preference command 66–67
bgp bestpath med missing-as-worst command 70
bgp confederation identifier command 64
bgp bestpath compare-routerid command 70–71
bgp confederation peers command 64
bgp bestpath med always command 70
bgp bestpath med confed command 70–71
bgp router submode 11
router bgp command 11
bgp session group submode 13
BGP update generation 25
BGP update groups example 111
bidirectional forwarding detection 4
Boolean operator precedence 447
Boolean operators, types 455

C

CIDR 414
Cisco IOS and Cisco IOS XR software differences, configuration 235
grouped 235
Cisco IOS XR OSPFv3 and OSPFv2 differences 302
clear policy 465
CLI (command-line interface) inheritance 302
CLI inheritance 302
community-set, inline set form 436
community-set, named set form 436
components 441
configuration 12, 14, 235, 241, 246, 248, 327, 344, 348
grouped configuration 235
grouping 12
inheriting 14
inheriting templates 14
Level 1 or Level 2 routing 246
MPLS TE 348
neighbors, nonbroadcast networks 327
single topology 248
SPF throttling 344
configuration and operation, verifying 366
configuration basics 444
configuration elements, editing 493
configuration, grouped 235
configurations (BGP) 14
configuring 105, 241, 256, 341, 352, 358
graceful restart 352
configuring (OSPFv2) 330
configuring authentication 262

configuring bgp additional paths 106
connected 518
control or prevent routing updates 423
controlling 253
controlling frequency 334
controlling the frequency 334
cost community, BGP 25
creating 335
customizing (IS-IS) 270
customizing routes 270

D

dampening 458
dampening (BFD) command 184
dampening, route 33
data structures in BGP and other protocols 402
debug 466
default 242, 307
  IS-IS 242
  OSPFv2 307
default address family 22, 40
default drop disposition 449
default originate 458, 479, 481, 483, 486
default route 307
default routes 242
default-accept-in 487
default-accept-out 488
default-information originate 491
defining 504
definitions 444
deploying 405
description 3, 24, 299, 312–313, 401
designate router (DR) 307
Designate Router (DR) 307
designated router (DR) 307
Designated Router (DR) 307
displaying information 354
disposition 452

E

echo disable command 183
echo ipv4 source command 177–178, 182
echo latency detect command 179
EIGRP 219
  authentication using keychain 219
EIGRP () 487
  policy attach points 487
    default-accept-in 487
EIGRP (Enhanced Interior Gateway Routing Protocol) 210, 216–218, 226, 487–489
  features 210
  hello interval and hold time 216
  overview 210
EIGRP (Enhanced Interior Gateway Routing Protocol) *(continued)*
  policy attach points 488–489
    default-accept-out 488
    if-policy-in 489
    if-policy-out 489
    policy-in 488
    policy-out 488
    redistribute 489
  policy attach points, default-accept-in 487
  restrictions 226
  routing policy options 218
  split horizon 216
  stub routing 217
elseif 454
enabling 246, 323
enforcing, BGP 23
show eigrp traffic command 228, 230
clear eigrp topology command 228–229
show isis mpls command 264–265
clear eigrp neighbors command" 228–229
show eigrp topology command 228, 230
show eigrp interfaces command 228–229
show rpl route-policy command 507
EXEC mode 18–21
  show bgp af-group command 19
  show bgp inheritance command 18
  show bgp neighbor command 18
  show bgp neighbor-group command 21
clear ospf command 367–368
show isis command 247–248
show isis spf-log command 268–269
edit command 507
export 463
extended community set, inline form 437
extended community set, named form 437

F

features 210
filter network updates 423
floating 520
functional overview 4, 300, 402

G

Generalized TTL Security Mechanism (GTSM), TTL value 319
Generalized TTL Security Mechanism (GTSM),configuring virtual links 319
global parameters 446
global-inbound 492
global-inbound, policy attach points 492
graceful restart 313, 352
graceful-restart helper command 314
graceful-restart interval command 314
graceful-restart lifetime command 314

grouped 235
grouped configuration 235
grouping 12

H

hello interval and hold time 216
hop count 414

I

if 454
if-policy-in 489
if-policy-out 489
implementing 431
  prerequisites 431
implementing in 431
  prerequisites 431
import 462
inheritance 14, 18
  configurations (BGP) 14
  monitoring 18
inheritance, monitoring 18
inheriting 14
inheriting templates 14
instance and router ID 305
inter-area-propagate 486
interface attributes and limitations 318
interface command 219–220, 323–324, 419–420
interface-inbound 492
interface-outbound 492
interior routers 305
IP fast reroute 244, 277
IP Fast Reroute 404
  loop-free alternate 404
IPv4 and IPv6 support 403
IPv6 237, 403
  IS-IS support 237
    single-topology 237
  RIB support 403
ipv6 checksum command 185–187
IPv6 prefixes over IPv4 session 9
IPv6 support 237
IS-IS 242
IS-IS (Intermediate System-to-Intermediate System) 235, 237–238, 241–242, 246, 248, 253, 256, 260, 262, 265, 268, 270, 272, 274–275, 277, 485–486
  adjacencies, tuning 265
  attached bit on an instance 242
  authentication, configuring 260
  Cisco IOS and Cisco IOS XR software differences, configuration 235
    grouped 235
  configuration 235, 246, 248
    grouped configuration 235
IS-IS (Intermediate System-to-Intermediate System) *(continued)*
  configuration *(continued)*
    Level 1 or Level 2 routing 246
    single topology 248
  configuration, grouped 235
  configuring authentication 262
  customizing routes 270
  default routes 242
  enabling 246
  grouped configuration 235
  IP fast reroute 277
  Level 1 or Level 2 routing, configuration 246
  LSP flooding 253
    controlling 253
  MPLS LDP IS-IS synchronization 272
  multi-instance IS-IS 241
  nonstop forwarding 238, 256
    configuring 256
  overload bit 241
    configuring 241
    on router 241
  policy attach points 485–486
    default originate 486
    inter-area-propagate 486
    redistribute 485
  priority for prefixes added to RIB 275
  set SPF interval 268
  single topology,configuring 248
  single topology,IPv6 support 237
  tagging IS-IS interface routes 274
IS-IS address family submode 235
IS-IS Overload Bit Avoidance 242
IS-IS support 237
  single-topology 237

K

key rollover 307
keychains 40

L

label consistency checker 404
LDP IS-IS synchronization 272
Level 1 or Level 2 routing 246
Level 1 or Level 2 routing, configuration 246
link-state advertisement (LSA) 307–308, 316
  OSPFv2 307
  OSPFv3 308, 316
load balancing 317
loop-free alternate 244, 404
LSA 307–308, 334, 339
  controlling frequency 334
  controlling the frequency 334
  on an OSPF ABR 339

LSA *(continued)*
types 307–308
LSP flooding 237, 253
controlling 253
on specific interfaces 237
lsp-check-interval command 253–254

M

maximum limits (BGP) 5
MD5 306
MD5 (OSPFv2) 306
MD5 authentication 306
Message Digest 5 (MD5) authentication 319
message statistics 403
metrics 414
modification 447
modifying 507
monitor 104
monitoring 18, 405
MPLS LDP IS-IS synchronization 272
MPLS TE 348
MPLS TE (Multiprotocol Label Switching traffic engineering)
configuring 348
OSPFv2 348
MPLS TE (Multiprotocol Label Switching traffic engineering), configuring for OSPFv2 348
MPLS TE, configuring 348
multi-area adjacency 318, 358
configuring 358
interface attributes and limitations 318
overview 318
multi-instance IS-IS 241
Multiprotocol Label Switching (MPLS) 272
LDP IS-IS synchronization 272
Multiprotocol Label Switching traffic engineering (MPLS TE), configuring fir IS-IS 263
multitopology 294
(example) 294

N

names 434
NBMA networks 306
neighbor address family command 12, 235
neighbor command 12, 235, 419–420
neighbor command (BFD) 165
neighbor command (OSPFv2 328–329
OSPFv3) 328–329
neighbor export 459
neighbor import 459
neighbor-group command 13
neighbor-orf 464
neighbors 5, 307
adjacency (OSPFv2) 307
neighbors *(continued)*
maximum limits (BGP) 5
neighbors, adjacency 307
neighbors, configuring nonbroadcast networks 327
neighbors, maximum limits on 5
neighbors, nonbroadcast networks 327
neighbors, nonbroadcast networks, configuring 327
network 460
next-hop 465
nonattached 493
nonstop forwarding 238, 256, 313, 346
configuring 256
description 313
not-so-stubby area 304

O

on an OSPF ABR 339
on router 241
on specific interfaces 237
OSPF 320, 368
queue tuning parameters 320
queue tuning parameters configuration 368
network command 328–329
virtual-link command 336–337
authentication message-digest command 336, 338
hello-interval (OSPF) command 328–329
default-cost command 325, 327
nssa command 325–326
dead interval command command 328–329
OSPFv2 307, 348
OSPFv2 (Open Shortest Path First Version 2) 299–300, 302, 305–307, 309, 312–313, 323, 325, 327, 330, 334–335, 339, 341, 344, 346, 348, 366, 460, 479, 481–482, 484
authentication, configuring 330
Cisco IOS XR OSPFv3 and OSPFv2 differences 302
CLI (command-line interface) inheritance 302
configuration 327, 348
MPLS TE 348
neighbors, nonbroadcast networks 327
configuration and operation, verifying 366
default route 307
description 299
designate router (DR) 307
Designate Router (DR) 307
enabling 323
functional overview 300
instance and router ID 305
LSA 307, 334, 339
controlling the frequency 334
on an OSPF ABR 339
types 307
MD5 authentication 306
MPLS TE, configuring 348
neighbors, adjacency 307
neighbors, nonbroadcast networks, configuring 327

OSPFv2 (Open Shortest Path First Version 2) *(continued)*
nonstop forwarding **313, 346**
description **313**
policy attach points **460, 479, 481–482, 484**
area-in **481**
area-out **482**
default originate **479, 481**
redistribute **460, 481, 484**
route authentication methods **306–307**
key rollover **307**
MD5 **306**
plain text **306**
strategies **306**
route redistribution **312, 341**
configuring **341**
description **312**
Shortest Path First (SPF) throttling **312**
description **312**
Shortest Path First (SPF) throttling,configuring **344**
SPF throttling, configuring **344**
stub and not-so-stubby area types, configuring **325**
supported OSPF network types **306**
NBMA networks **306**
point to point networks **306**
virtual link **309, 335**
creating **335**
transit area **309**
OSPFv3 **308, 313, 316**
graceful restart **313**
OSPFv3 (Open Shortest Path First Version 3) **299–300, 302, 305, 307–309, 316–317, 323, 325, 327, 334, 339, 341, 344, 352, 366, 460, 481, 483–484**
Cisco IOS XR OSPFv3 and OSPFv2 differences **302**
CLI inheritance **302**
configuration **344**
SPF throttling **344**
configuration and operation, verifying **366**
configuring **352**
graceful restart **352**
default route **307**
description **299**
enabling **323**
functional overview **300**
instance and router ID **305**
link-state advertisement (LSA) **316**
load balancing **317**
LSA **308, 334, 339**
controlling frequency **334**
on an OSPF ABR **339**
types **308**
neighbors, configuring nonbroadcast networks **327**
policy attach points **460, 481, 483–484**
default originate **483**
redistribute **460, 481, 484**
routes, redistribute **341**
SPF (Shortest Path First) throttling configuring **344**
OSPFv3 (Open Shortest Path First Version 3) *(continued)*
stub and not-so-stubby area types, configuring **325**
virtual link, description **309**
OSPFv3 Graceful Restart feature **313, 316, 354**
adjacency **316**
displaying information **354**
OSPFv3 SPF **310**
nsf command **257, 421–422**
default-information originate command **270–271**
nsf interval command **347–348**
timers throttle spf command **344–345, 364–365**
mpls traffic-eng router-id command **349**
mpls traffic-eng area command **349–350**
overload bit **241**
configuration **241**
configuring **241**
on router **241**
overview **210, 318, 433**

P

parameterization **445**
PCE extensions to OSPFv2 **319**
plain text **306**
point to point networks **306**
point-to-point networks **306**
policy **444–445, 447, 449–451**
attributes **445, 447**
Boolean operator precedence **447**
configuration basics **444**
default drop disposition **449**
definitions **444**
statement processing **449**
statements, types **451**
verification **450**
policy attach points **458–466, 479, 481–489, 491–492**
allocate label **464**
area-in **481**
area-out **482**
clear policy **465**
dampening **458**
debug **466**
default originate **458, 479, 481, 483, 486**
default-accept-in **487**
default-accept-out **488**
default-information originate **491**
export **463**
global-inbound **492**
if-policy-in **489**
if-policy-out **489**
import **462**
inter-area-propagate **486**
interface-inbound **492**
interface-outbound **492**
neighbor export **459**
neighbor import **459**

policy attach points *(continued)*
neighbor-orf 464
network 460
next-hop 465
policy-in 488
policy-out 488
redistribute 460, 481, 484–485, 489, 491
retain route target 464
show bgp 461
policy attach points, default-accept-in 487
policy attributes 445, 447
modification 447
parameterization 445
policy-in 488
policy-out 488
policy, modifying 492–493
attached 492
nonattached 493
prefix prioritization 310
OSPFv2 SPF 310
prefix-set 439
prerequisites 401, 431
priority for prefixes added to RIB 275

Q

quarantining 404
queue tuning parameters 320
queue tuning parameters configuration 368

R

redistribute 460, 481, 484–485, 489, 491
redistribute (OSPFv2, OSPFv3) 341
redistributing (OSPFv2, OSPFv3) 341
redistribution 415
remark 451
remote-as command (BFD) 165
restrictions 226
retain route target 464
RFC 2328, OSPF Version 2 301
RFC 2453, RIP Version 2 414
RFC 2740 OSPFv3 301
RFC 3682 319
Generalized TTL Security Mechanism (GTSM), TTL value 319
RIB 403–404
quarantining 404
statistics 403
RIB (Routing Information Base) 401–403, 405, 408
administrative distance 402
data structures in BGP and other protocols 402
deploying 405
description 401
examples 408
functional overview 402
RIB (Routing Information Base) *(continued)*
IPv4 and IPv6 support 403
monitoring 405
prerequisites 401
RIB quarantining 404
RIB statistics 403
RIB support 403
RIP (Routing Information Protocol) 492
global-inbound, policy attach points 492
policy attach points 492
global-inbound 492
RIP (Routing Information Protocol) 414–417, 423, 425, 491–492
administrative distance 416
bandwidth 423
benefits 414
CIDR 414
control or prevent routing updates 423
filter network updates 423
hop count 414
metrics 414
policy attach points 491–492
default-information originate 491
global-inbound 492
interface-inbound 492
interface-outbound 492
redistribute 491
redistribution 415
route policy creation 425
route timers 415
routing loops 423
routing policy options 417
split horizon, enabling IP 415
VLSMs (variable-length subnet masks) 414
WAN link 423
RIP v2 supported features 414
route authentication methods 306–307
key rollover 307
MD5 306
plain text 306
strategies 306
route consistency checker 404
route dampening 33
end-policy command 60–61
route-policy command 60, 85, 221–222, 225, 423–425, 504–505
route policy creation 425
route redistribution 312, 341
configuring 341
description 312
route redistribution (OSPFv2, OSPFv3) 312
route reflectors 34
route timers 415
set eigrp-metric command 225
route-policy pass-all command 23
route, key rollover (OSPFv2) 307
range command 339–340
distance command 219–220

stub command 227, 325–326
net command 246–247
is-type command 246–247
area command 323–324
router bgp command 11, 164–165
router bgp neighbor group address family configuration mode,address family command 13
default-metric command 219–220
log-neighbor-changes command 228–229
redistribute command 223
log-neighbor-warnings command 228–229
redistribute maximum-prefix 223
maximum-prefix command 223–224
maximum paths command 223–224
timers nsf route-hold command 223–224
router eigrp command 219–220
holdtime command 219–220
summary-address command 221–222
bandwidth-percent command 219–220
router identifier 4
redistribute isis command 270–271
single-topology command 249, 251
metric-style wide command 274–276
spf-interval command 268–269
attached-bit send command 270, 272
mpls traffic-eng command 264
spf prefix-priority command 276
sp-refresh-interval command 253–254
lsp-mtu command 253–254
ignore-lsp-errors command 253–254
max-lsp-lifetime command 253–254
lsp-gen-interval command 253–254
address-family (IS-IS) command 249, 251, 291
nsf interface-timer command 257
nsf interface-expires command 257
lsp-password command 260
retransmit-interval command 253, 255
csnp-interval command 253, 255
hello-password command 260–261, 266–267
hello-multiplier command 266–267
mesh-group command 253, 255
hello-padding command 266
hello-interval (IS-IS) command 266–267
retransmit-throttle-interval command 253, 255
tag command 274–275
router isis interface configuration submode 249–250
router ospf command 323–324
router ospf command (BFD) 166–167
summary-prefix command 341, 343
See also router ospfv3 configuration submode
message-digest-key command 331–332
authentication command (OSPFv2) 331–332
router-id command 323–324
log adjacency changes command 323–324, 364–365
timers lsa min-interval command 334–335
timers lsa group-pacing command 334–335
timers lsa refresh command 334–335
timers lsa gen-interval command 334–335
router ospfv3 command 323–324
router ospfv3 command (BFD) 168–169
router rib command 406
address-family command 406
router rip command 419–420
split-horizon disable command 421–422
metric-zero-accept command 421–422
output-delay command 421–422
auto-summary command 421
poison-reverse command 421–422
broadcast-for-v2 command 419–420
receive version command 419–420
timers basic command 421
send version command 419–420
passive-interface command 423–424
routes 242, 270, 295, 307, 341
  customizing (IS-IS) 270
  default 242, 307
    IS-IS 242
    OSPFv2 307
  redistribute (OSPFv2, OSPFv3) 341
  redistribute IS-IS routes (example) 295
  redistribute IS-IS routes example 295
  redistributing (OSPFv2, OSPFv3) 341
routes, redistribute 341
routing components 303–305, 307
  ABRs (area border routers) 304
  Area Border Routers (ABRs) 304
  ASBRs (autonomous system boundary routers) 305
  autonomous systems 303
  backbone area 304
  designated router (DR) 307
  Designated Router (DR) 307
  interior routers 305
  not-so-stubby area 304
  stub area 304
routing domain confederation 34
routing loops 423
routing policy 23, 431, 451–452, 454, 493, 504–505, 507–509
  attaching to BGP neighbor 505
  configuration elements, editing 493
  defining 504
  defining (example) 508
  enforcing, BGP 23
  implementing 431
    prerequisites 431
  implementing in 431
    prerequisites 431
  inbound (example) 508
  modifying 507
  modular inbound (example) 509
  statements 451–452, 454
    action 454
    disposition 452

routing policy *(continued)*
statements *(continued)*
elseif **454**
if **454**
remark **451**
Routing Policy **432**
restrictions, configuring routing policy **432**
routing policy options **218, 417**
routing policy, enforcing **23**
RPL (routing policy language) **433–437, 439, 441, 444–445, 447, 449–451, 455**
Boolean operators, types **455**
components **441**
overview **433**
policy **444–445, 447, 449–451**
attributes **445, 447**
Boolean operator precedence **447**
configuration basics **444**
default drop disposition **449**
definitions **444**
statement processing **449**
statements, types **451**
verification **450**
policy attributes **445, 447**
modification **447**
parameterization **445**
structure **434–437, 439**
as-path-set, inline set form **435**
as-path-set, named set form **435**
community-set, inline set form **436**
community-set, named set form **436**
extended community set, inline form **437**
extended community set, named form **437**
names **434**
prefix-set **439**
sets **434**

S

security ttl command **364–365**
set SPF interval **268**
set-overload-bit command **270**
sets **434**
Shortest Path First (SPF) throttling **312**
description **312**
Shortest Path First (SPF) throttling, configuring **344**
show bgp **461**
show bgp af-group command **19**
show bgp cidr-only command **102**
show bgp command **102**
show bgp community command **102**
show bgp inheritance command **18**
show bgp ipv4 unicast summary command **103–104**
show bgp neighbor command **18**
show bgp neighbor-group command **21, 102–103**
show bgp neighbors command **102**
show bgp paths command **102–103**
show bgp process command **103**
show bgp process detail command **103–104**
show bgp session-group command **20**
show bgp summary command **102–104**
show eigrp neighbors command **227–229**
show isis database command **253, 256**
show isis database-log command **253, 256**
show isis interface command **266, 268**
show isis lsp-log command **253, 256**
show isis mpls traffic-eng adjacency-log command **264–265**
show isis mpls traffic-eng advertisements command **264–265**
show isis neighbors command **266, 268**
show isis topology command **249, 252**
show ospf command **336–337**
show ospfv3 command **336–337**
show placement program bgp command **103–104**
show placement program brib command **103–104**
show protocols eigrp command **228, 230**
show running-config command **257–258**
single topology **248**
single topology,configuring **248**
single topology,IPv6 support **237**
single-topology **237, 268, 294**
configuring (example) **294**
IPv6 support **237**
set SPF interval **268**
specified **519**
SPF (Shortest Path First) throttling configuring **344**
SPF throttling **344**
SPF throttling, configuring **344**
OSPFv2 (Open Shortest Path First Version 2) **344**
split horizon **216**
split horizon, enabling IP **415**
statement processing **449**
statements **451–452, 454**
action **454**
disposition **452**
elseif **454**
if **454**
remark **451**
statements, types **451**
static routes **518–520**
administrative distance **518**
connected **518**
floating **520**
specified **519**
statistics **403**
strategies **306**
structure **434–437, 439**
as-path-set, inline set form **435**
as-path-set, named set form **435**
community-set, inline set form **436**
community-set, named set form **436**
extended community set, inline form **437**
extended community set, named form **437**

structure *(continued)*
names 434
prefix-set 439
sets 434
stub and not-so-stubby area types, configuring 325
stub area 304
stub area types, configuring (OSPFv3) 325
stub routing 217
router isis address family submode 270–271
supported OSPF network types 306
NBMA networks 306
point to point networks 306

T

tagging IS-IS interface routes 274
transit area 309
transit area (OSPFv2) 309
types 307–308

U

update groups 24–25, 104
BGP configuration 25
BGP update generation 25
description 24
monitor 104

V

verification 450
virtual link 309, 335
creating 335
transit area 309
transit area (OSPFv2) 309
virtual link, description 309
VLSMs (variable-length subnet masks) 414

W

WAN link 423