CONTENTS

Preface xxiii

Obtaining Documentation and Submitting a Service Request xxiii

CHAPTER 1 New and Changed Routing Features 1

CHAPTER 2 Implementing BGP 5

Prerequisites for Implementing BGP 7

Information About Implementing BGP 7

BGP Functional Overview 7

BGP Router Identifier 8

BGP Default Limits 9

BGP Next Hop Tracking 10

Scoped IPv4/VPNv4 Table Walk 11

Reordered Address Family Processing 11

New Thread for Next-Hop Processing 11

show, clear, and debug Commands 11

Autonomous System Number Formats in BGP 12

2-byte Autonomous System Number Format 12

4-byte Autonomous System Number Format 12

as-format Command 12

BGP Configuration 12

Configuration Modes 13

Router Configuration Mode 13

Router Address Family Configuration Mode 13

Neighbor Configuration Mode 13

Neighbor Address Family Configuration Mode 13

VRF Configuration Mode 13
Contents

Multiprotocol BGP 41
Route Dampening 43
  Minimizing Flapping 44
BGP Routing Domain Confederation 44
BGP Route Reflectors 44
Default Address Family for show Commands 48
MPLS VPN Carrier Supporting Carrier 48
BGP Keychains 49
BGP Nonstop Routing 49
BGP Prefix Independent Convergence Unipath Primary/Backup 50
BGP Local Label Retention 51
Command Line Interface (CLI) Consistency for BGP Commands 51
BGP Additional Paths 51
iBGP Multipath Load Sharing 52
Accumulated Interior Gateway Protocol Attribute 52
Per VRF and Per CE Label for IPv6 Provider Edge 53
IPv4 BGP-Policy Accounting on Cisco ASR 9000's A9K-SIP-700 53
IPv6 Unicast Routing on Cisco ASR 9000's A9K-SIP-700 53
IPv6 uRPF Support on Cisco ASR 9000's A9K-SIP-700 54
Constrained Route Distribution for BGP/MPLS Internet Protocol VPNs 54
  Constrained Route Distribution Benefits 54
  BGP RT-constrain SAFI—rt-filter 55
Selective VRF Download 55
  Line Card Roles and Filters in Selective VRF Download 56
BGP Accept Own 56
BGP DMZ Link Bandwidth for Unequal Cost Recursive Load Balancing 58
BFD Multihop Support for BGP 58
BGP Multi-Instance and Multi-AS 58
BGP Prefix Origin Validation Based on RPKI 59
  Configuring RPKI Cache-server 59
  Configuring RPKI Prefix Validation 62
  Configuring RPKI Bestpath Computation 63
BGP 3107 PIC Updates for Global Prefixes 64
BGP Prefix Independent Convergence for RIB and FIB 65
BGP Update Message Error Handling 65
Contents

BGP Attribute Filtering 66

BGP Attribute Filter Actions 66

BGP Error Handling and Attribute Filtering Syslog Messages 66

How to Implement BGP 67

Enabling BGP Routing 67

Configuring Multiple BGP Instances for a Specific Autonomous System 69

Configuring a Routing Domain Confederation for BGP 70

Resetting an eBGP Session Immediately Upon Link Failure 71

Logging Neighbor Changes 72

Adjusting BGP Timers 72

Changing the BGP Default Local Preference Value 73

Configuring the MED Metric for BGP 74

Configuring BGP Weights 74

Tuning the BGP Best-Path Calculation 76

Indicating BGP Back-door Routes 77

Configuring Aggregate Addresses 78

Redistributing iBGP Routes into IGP 79

Redistributing Prefixes into Multiprotocol BGP 80

Configuring BGP Route Dampening 82

Applying Policy When Updating the Routing Table 87

Setting BGP Administrative Distance 88

Configuring a BGP Neighbor Group and Neighbors 88

Configuring a Route Reflector for BGP 91

Configuring BGP Route Filtering by Route Policy 92

Configuring BGP Attribute Filtering 94

Configuring BGP Next-Hop Trigger Delay 95

Disabling Next-Hop Processing on BGP Updates 96

Configuring BGP Community and Extended-Community Advertisements 97

Configuring the BGP Cost Community 99

Configuring Software to Store Updates from a Neighbor 102

Configuring a VPN Routing and Forwarding Instance in BGP 104

Defining Virtual Routing and Forwarding Tables in Provider Edge Routers 104

Configuring the Route Distinguisher 106

Configuring PE-PE or PE-RR Interior BGP Sessions 107
Contents

Configuring Route Reflector to Hold Routes That Have a Defined Set of RT Communities 110
Configuring BGP as a PE-CE Protocol 111
Redistribution of IGPs to BGP 115
Configuring Keychains for BGP 117
Disabling a BGP Neighbor 118
Resetting Neighbors Using BGP Inbound Soft Reset 119
Resetting Neighbors Using BGP Outbound Soft Reset 120
Resetting Neighbors Using BGP Hard Reset 121
Clearing Caches, Tables, and Databases 122
Displaying System and Network Statistics 123
Displaying BGP Process Information 124
Monitoring BGP Update Groups 126
Configuring BGP Nonstop Routing 127
Installing Primary Backup Path for Prefix Independent Convergence (PIC) 128
Retaining Allocated Local Label for Primary Path 129
Configuring BGP Additional Paths 130
Configuring iBGP Multipath Load Sharing 131
Originating Prefixes with AiGP 132
Configuring BGP Accept Own 134
Enabling BGP Unequal Cost Recursive Load Balancing 135
Configuration Examples for Implementing BGP 137
   Enabling BGP: Example 137
   Displaying BGP Update Groups: Example 138
   BGP Neighbor Configuration: Example 139
   BGP Confederation: Example 139
   BGP Route Reflector: Example 141
   BGP Nonstop Routing Configuration: Example 141
   Primary Backup Path Installation: Example 141
   Allocated Local Label Retention: Example 142
   iBGP Multipath Loadsharing Configuration: Example 142
   Originating Prefixes With AiGP: Example 142
   BGP Accept Own Configuration: Example 142
   BGP Unequal Cost Recursive Load Balancing: Example 143
Where to Go Next 145
## Chapter 3: Implementing BFD

- **Prerequisites for Implementing BFD** 151
- **Restrictions for Implementing BFD** 151
- **Information About BFD** 152
  - Differences in BFD in Cisco IOS XR Software and Cisco IOS Software 152
  - BFD Multipath Sessions Support on nV Edge System 153
  - BFD Modes of Operation 153
  - BFD Packet Information 154
    - BFD Source and Destination Ports 154
    - BFD Packet Intervals and Failure Detection 154
      - BFD Packet Intervals on Physical Interfaces 154
      - BFD Packet Intervals on Bundle Member Links 155
      - Control Packet Failure Detection In Asynchronous Mode 155
      - Echo Packet Failure Detection In Asynchronous Mode 155
      - Echo Failure Detection Examples 156
      - Summary of Packet Intervals and Failure Detection Times for BFD on Bundle Interfaces 157
    - Echo Packet Latency 158
  - Priority Settings for BFD Packets 158
  - BFD for IPv4 159
    - Enabling BFD on a Static Route 160
  - BFD for IPv6 162
  - BFD on Bundled VLANs 162
  - BFD Over Member Links on Link Bundles 163
    - Overview of BFD State Change Behavior on Member Links and Bundle Status 163
  - BFD Multipath Sessions 165
    - BFD for MultiHop Paths 165
    - Setting up BFD Multihop 166
    - BFD over Satellite Interfaces 166
  - How to Configure BFD 166
    - BFD Configuration Guidelines 166
    - Configuring BFD Under a Dynamic Routing Protocol or Using a Static Route 167
    - Enabling BFD on a BGP Neighbor 167
Enabling BFD for OSPF on an Interface 168
Enabling BFD for OSPFv3 on an Interface 170
Enabling BFD on a Static Route 172
Configuring BFD on Bundle Member Links 173
Prerequisites for Configuring BFD on Bundle Member Links 173
Specifying the BFD Destination Address on a Bundle 174
Enabling BFD Sessions on Bundle Members 174
Configuring the Minimum Thresholds for Maintaining an Active Bundle 175
Configuring BFD Packet Transmission Intervals and Failure Detection Times on a Bundle 176
Configuring Allowable Delays for BFD State Change Notifications Using Timers on a Bundle 178
Enabling Echo Mode to Test the Forwarding Path to a BFD Peer 179
Overriding the Default Echo Packet Source Address 179
Specifying the Echo Packet Source Address Globally for BFD 179
Specifying the Echo Packet Source Address on an Individual Interface or Bundle 180
Configuring BFD Session Teardown Based on Echo Latency Detection 181
Delaying BFD Session Startup Until Verification of Echo Path and Latency 182
Disabling Echo Mode 183
Disabling Echo Mode on a Router 183
Disabling Echo Mode on an Individual Interface or Bundle 184
Minimizing BFD Session Flapping Using BFD Dampening 185
Enabling and Disabling IPv6 Checksum Support 186
Enabling and Disabling IPv6 Checksum Calculations for BFD on a Router 186
Enabling and Disabling IPv6 Checksum Calculations for BFD on an Individual Interface or Bundle 187
Clearing and Displaying BFD Counters 188
Configuration Examples for Configuring BFD 189
BFD Over BGP: Example 189
BFD Over OSPF: Examples 189
BFD Over Static Routes: Examples 190
BFD on Bundled VLANs: Example 190
BFD on Bundle Member Links: Examples 191
Echo Packet Source Address: Examples 192
Echo Latency Detection: Examples 192
CHAPTER 4

Implementing EIGRP 197

Prerequisites for Implementing EIGRP 198
Restrictions for Implementing EIGRP 198
Information About Implementing EIGRP 198
  EIGRP Functional Overview 198
  EIGRP Features 199
  EIGRP Components 199
  EIGRP Configuration Grouping 200
  EIGRP Configuration Modes 200
  EIGRP Interfaces 201
Redistribution for an EIGRP Process 201
Metric Weights for EIGR PRouting 202
  Mismatched K Values 202
  Goodbye Message 203
Percentage of Link Bandwidth Used for EIGRP Packets 203
Floating Summary Routes for an EIGRP Process 203
Split Horizon for an EIGRP Process 205
Adjustment of Hello Interval and Hold Time for an EIGRP Process 206
Stub Routing for an EIGRP Process 206
Route Policy Options for an EIGRP Process 207
EIGRP Layer 3 VPN PE-CE Site-of-Origin 208
  Router Interoperation with the Site-of-Origin Extended Community 208
Route Manipulation using SoO match condition 209
EIGRP v4/v6 Authentication Using Keychain 210
How to Implement EIGRP 210
  Enabling EIGRP Routing 211
  Configuring Route Summarization for an EIGRP Process 212
  Redistributing Routes for EIGRP 214
  Creating a Route Policy and Attaching It to an EIGRP Process 216
  Configuring Stub Routing for an EIGRP Process 217
  Configuring EIGRP as a PE-CE Protocol 218
  Redistributing BGP Routes into EIGRP 220
  Monitoring EIGRP Routing 222
  Configuring an EIGRP Authentication Keychain 224
    Configuring an Authentication Keychain for an IPv4/IPv6 Interface on a Default VRF 224
    Configuring an Authentication Keychain for an IPv4/IPv6 Interface on a Nondefault VRF 225
Configuration Examples for Implementing EIGRP 226
  Configuring a Basic EIGRP Configuration: Example 226
  Configuring an EIGRP Stub Operation: Example 227
  Configuring an EIGRP PE-CE Configuration with Prefix-Limits: Example 227
  Configuring an EIGRP Authentication Keychain: Example 228
Additional References 228

CHAPTER 5
Implementing IS-IS 231
  Prerequisites for Implementing IS-IS 231
  Restrictions for Implementing IS-IS 231
  Information About Implementing IS-IS 232
    IS-IS Functional Overview 232
    Key Features Supported in the Cisco IOS XR IS-IS Implementation 232
  IS-IS Configuration Grouping 233
  IS-IS Configuration Modes 233
    Router Configuration Mode 233
    Router Address Family Configuration Mode 233
    Interface Configuration Mode 233
    Interface Address Family Configuration Mode 234
  IS-IS Interfaces 234
Multitopology Configuration 234
IPv6 Routing and Configuring IPv6 Addressing 234
Limit LSP Flooding 235
   Flood Blocking on Specific Interfaces 235
   Mesh Group Configuration 235
Maximum LSP Lifetime and Refresh Interval 235
Single-Topology IPv6 Support 236
Multitopology IPv6 for IS-IS 236
IS-IS Authentication 236
Nonstop Forwarding 237
Multi-Instance IS-IS 238
Multiprotocol Label Switching Traffic Engineering 238
Overload Bit on Router 238
Overload Bit Configuration During Multitopology Operation 239
IS-IS Overload Bit Avoidance 239
Default Routes 239
Attached Bit on an IS-IS Instance 239
IS-IS Support for Route Tags 240
Multicast-Intact Feature 240
Multicast Topology Support Using IS-IS 240
MPLS Label Distribution Protocol IGP Synchronization 241
   MPLS LDP-IGP Synchronization Compatibility with LDP Graceful Restart 241
   MPLS LDP-IGP Synchronization Compatibility with IGP Nonstop Forwarding 241
Label Distribution Protocol IGP Auto-configuration 242
MPLS TE Forwarding Adjacency 242
MPLS TE Interarea Tunnels 242
IP Fast Reroute 242
Unequal Cost Multipath Load-balancing for IS-IS 243
Enabling IS-IS and Configuring Level 1 or Level 2 Routing 243
Configuring Single Topology for IS-IS 245
Configuring Multitopology Routing 249
   Restrictions for Configuring Multitopology Routing 249
   Information About Multitopology Routing 249
   Configuring a Global Topology and Associating It with an Interface 250
   Enabling an IS-IS Topology 251
CHAPTER 6

Implementing OSPF 289

Prerequisites for Implementing OSPF 290

Information About Implementing OSPF 291

OSPF Functional Overview 291

Key Features Supported in the Cisco IOS XR Software OSPF Implementation 292

Comparison of Cisco IOS XR Software OSPFv3 and OSPFv2 292
OSPF Hierarchical CLI and CLI Inheritance 293
OSPF Routing Components 293
  Autonomous Systems 294
  Areas 294
    Backbone Area 295
    Stub Area 295
    Not-so-Stubby Area 295
  Routers 295
    Area Border Routers 295
    Autonomous System Boundary Routers (ASBR) 296
    Interior Routers 296
OSPF Process and Router ID 296
Supported OSPF Network Types 297
Route Authentication Methods for OSPF 297
  Plain Text Authentication 297
  MD5 Authentication 297
  Authentication Strategies 297
  Key Rollover 298
Neighbors and Adjacency for OSPF 298
Designated Router (DR) for OSPF 298
Default Route for OSPF 298
Link-State Advertisement Types for OSPF Version 2 299
Link-State Advertisement Types for OSPFv3 299
Virtual Link and Transit Area for OSPF 301
Passive Interface 301
OSPFv2 Sham Link Support for MPLS VPN 302
OSPF SPF Prefix Prioritization 303
Route Redistribution for OSPF 305
OSPF Shortest Path First Throttling 305
Nonstop Forwarding for OSPF Version 2 306
Graceful Shutdown for OSPFv3 306
  Modes of Graceful Restart Operation 307
    Restart Mode 307
    Helper Mode 307
  Graceful Restart Requirements and Restrictions 308
Configuring OSPF as a Provider Edge to Customer Edge (PE-CE) Protocol 353
Creating Multiple OSPF Instances (OSPF Process and a VRF) 355
Configuring Multi-area Adjacency 356
Configuring Label Distribution Protocol IGP Auto-configuration for OSPF 358
Configuring LDP IGP Synchronization: OSPF 359
Configuring Authentication Message Digest Management for OSPF 360
Examples 361
Configuring Generalized TTL Security Mechanism (GTSM) for OSPF 362
Examples 364
Verifying OSPF Configuration and Operation 364
Configuring IP Fast Reroute Loop-free Alternate 367
   Enabling IPFRR LFA 367
   Excluding an Interface From IP Fast Reroute Per-link Computation 368
   Enabling OSPF Interaction with SRMS Server 368
Configuration Examples for Implementing OSPF 370
   Cisco IOS XR Software for OSPF Version 2 Configuration: Example 370
   CLI Inheritance and Precedence for OSPF Version 2: Example 371
   MPLS TE for OSPF Version 2: Example 372
   ABR with Summarization for OSPFv3: Example 372
   ABR Stub Area for OSPFv3: Example 373
   ABR Totally Stub Area for OSPFv3: Example 373
   Configuring OSPF SPF Prefix Prioritization: Example 373
   Route Redistribution for OSPFv3: Example 374
   Virtual Link Configured Through Area 1 for OSPFv3: Example 374
   Virtual Link Configured with MD5 Authentication for OSPF Version 2: Example 375
   VPN Backbone and Sham Link Configured for OSPF Version 2: Example 376
Where to Go Next 377
Additional References 377

CHAPTER 7
Implementing and Monitoring RIB 381
   Prerequisites for Implementing RIB 382
   Information About RIB Configuration 382
      Overview of RIB 382
      RIB Data Structures in BGP and Other Protocols 382
      RIB Administrative Distance 383
CHAPTER 8 Implementing RIP 399

Prerequisites for Implementing RIP 400
Information About Implementing RIP 400
RIP Functional Overview 400
Split Horizon for RIP 401
Route Timers for RIP 401
Route Redistribution for RIP 402
Default Administrative Distances for RIP 402
Routing Policy Options for RIP 403
Authentication Using Keychain in RIP 403
CHAPTER 9
Implementing Routing Policy 421

Prerequisites for Implementing Routing Policy 422
Restrictions for Implementing Routing Policy 422
Information About Implementing Routing Policy 423
Routing Policy Language 423
Routing Policy Language Overview 423
Routing Policy Language Structure 423

Names 424
Sets 424
as-path-set 425
community-set 426
extcommunity-set 427
prefix-set 430
rd-set 431
Routing Policy Language Components 431
Routing Policy Language Usage 432
Routing Policy Configuration Basics 434
Policy Definitions 434
Parameterization 435
Parameterization at Attach Points 435
Global Parameterization 436
Semantics of Policy Application 436
Boolean Operator Precedence 437
Multiple Modifications of the Same Attribute 437
When Attributes Are Modified 438
Default Drop Disposition 438
Control Flow 439
Policy Verification 439
Range Checking 439
Incomplete Policy and Set References 440
Attached Policy Modification 440
Verification of Attribute Comparisons and Actions 440
Policy Statements 441
Remark 441
Disposition 441
Action 443
If 443
Boolean Conditions 444
apply 445
Attach Points 446
BGP Policy Attach Points 446
Additional-Path 446
Aggregation 447
Dampening 448
Default Originate 448
Neighbor Export 449
Neighbor Import 449
Network 450
Contents

If-Policy-In 479
If-Policy-Out 479
Redistribute 479
EIGRP Attributes and Operators 480

RIP Policy Attach Points 481
   Default-Information Originate 481
   Redistribute 481
   Global-Inbound 482
   Global-Outbound 482
   Interface-Inbound 482
   Interface-Outbound 482
   RIP Attributes and Operators 482

PIM Policy Attach Points 484

Attached Policy Modification 484
Nonattached Policy Modification 484

Editing Routing Policy Configuration Elements 484
   Editing Routing Policy Configuration Elements Using the Nano Editor 485
   Editing Routing Policy Configuration Elements Using the Emacs Editor 485
   Editing Routing Policy Configuration Elements Using the Vim Editor 486
   Editing Routing Policy Configuration Elements Using CLI 486
   Editing Routing Policy Language set elements Using XML 486

Hierarchical Policy Conditions 487
   Apply Condition Policies 487
   Behavior of pass/drop/done RPL Statements for Simple Hierarchical Policies 488
   Behavior of pass/drop/done RPL Statements for Hierarchical Policy Conditions 488

Nested Wildcard Apply Policy 489
Wildcards for Route Policy Sets 490

How to Implement Routing Policy 490
   Defining a Route Policy 490
   Attaching a Routing Policy to a BGP Neighbor 491
   Modifying a Routing Policy Using a Text Editor 492

Configuration Examples for Implementing Routing Policy 493
   Routing Policy Definition: Example 493
   Simple Inbound Policy: Example 494
   Modular Inbound Policy: Example 495
CHAPTER 10 
Implementing Static Routes 501
- Prerequisites for Implementing Static Routes 502
- Restrictions for Implementing Static Routes 502
- Information About Implementing Static Routes 502
  - Static Route Functional Overview 502
  - Default Administrative Distance 502
  - Directly Connected Routes 503
  - Recursive Static Routes 503
  - Fully Specified Static Routes 504
  - Floating Static Routes 504
  - Default VRF 504
  - IPv4 and IPv6 Static VRF Routes 504
  - Dynamic ECMP 505
- How to Implement Static Routes 505
  - Configure Static Route 505
  - Configure Floating Static Route 506
  - Configure Static Routes Between PE-CE Routers 508
  - Change Maximum Number of Allowable Static Routes 509
  - Associate VRF with a Static Route 510
  - Enabling Object Tracking for Static Routes 511
- Configuration Examples 512
  - Configuring Traffic Discard: Example 513
  - Configuring a Fixed Default Route: Example 513
  - Configuring a Floating Static Route: Example 513
  - Configuring a Static Route Between PE-CE Routers: Example 513
- Additional References 513

CHAPTER 11 
Implementing RCMD 517
- Route Convergence Monitoring and Diagnostics 517
- Configuring Route Convergence Monitoring and Diagnostics 518
Preface

From Release 6.1.1 onwards, Cisco introduces support for the 64-bit Linux-based IOS XR operating system. Extensive feature parity is maintained between the 32-bit and 64-bit environments. Unless explicitly marked otherwise, the contents of this document are applicable for both the environments. For more details on Cisco IOS XR 64 bit, refer to the Release Notes for Cisco ASR 9000 Series Routers, Release 6.1.1 document.

The Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide preface contains these sections:

- Obtaining Documentation and Submitting a Service Request, page xxiii

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see What's New in Cisco Product Documentation.

To receive new and revised Cisco technical content directly to your desktop, you can subscribe to the What's New in Cisco Product Documentation RSS feed. RSS feeds are a free service.
New and Changed Routing Features

This table summarizes the new and changed feature information for the Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, and tells you where they are documented.

- New and Changed Features, page 1

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Introduced/Changed in Release</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP Attribute Filtering</td>
<td>This feature was introduced.</td>
<td>Release 4.2.3</td>
<td>Implementing BGP chapter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- BGP Attribute Filtering, on page 66</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- BGP Attribute Filter Actions, on page 66</td>
</tr>
<tr>
<td>BGP Error Handling and Attribute Filtering Syslog Messages</td>
<td>The BGP error handling syslog messages were enhanced with BGP attribute filtering keywords.</td>
<td>Release 4.2.3</td>
<td>Implementing BGP chapter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BGP Error Handling and Attribute Filtering Syslog Messages, on page 66</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Introduced/Changed in Release</td>
<td>Where Documented</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------------------------------------</td>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BGP Prefix Origin Validation Based on RPKI</td>
<td>This feature was introduced.</td>
<td>Release 4.2.1</td>
<td>Implementing BGP chapter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• BGP Prefix Origin Validation Based on RPKI, on page 59</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Configuring RPKI Cache-server, on page 59</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Configuring RPKI Prefix Validation, on page 62</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Configuring RPKI Bestpath Computation, on page 63</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Refer BGP Commands chapter in Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference for information on the commands used for configuring and verifying BGP Prefix Origin Validation.</td>
</tr>
<tr>
<td>BGP 3107 PIC Updates for Global Prefixes</td>
<td>This feature was introduced.</td>
<td>Release 4.2.1</td>
<td>Implementing BGP chapter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BGP 3107 PIC Updates for Global Prefixes, on page 64</td>
</tr>
<tr>
<td>BGP Prefix Independent Convergence for RIB and FIB</td>
<td>This feature was introduced.</td>
<td>Release 4.2.1</td>
<td>Implementing BGP chapter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BGP Prefix Independent Convergence for RIB and FIB, on page 65</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Introduced/Changed in Release</td>
<td>Where Documented</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------------------------------</td>
<td>-------------------</td>
</tr>
</tbody>
</table>
| OSPF SPF Prefix Prioritization | This feature was supported on OSPFv3 protocol. | Release 4.2.1 | Implementing OSPF chapter:  
  - OSPF SPF Prefix Prioritization, on page 303  
  - Configuring OSPF SPF Prefix Prioritization, on page 349  
  - Configuring OSPF SPF Prefix Prioritization: Example, on page 373  
Refer OSPFv3 Commands chapter in Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference for information on the commands used for configuring and verifying OSPFv3 SPF Prefix Prioritization. |
| Management Information Base (MIB) for OSPFv3 | This feature was introduced. | Release 4.2.1 | Implementing OSPF chapter  
  Management Information Base (MIB) for OSPFv3, on page 313 |
| Nested Wildcard Apply Policy | This feature was introduced. | Release 4.2.1 | Implementing Routing Policy chapter  
Nested Wildcard Apply Policy, on page 489  
Refer apply command in Routing Policy Language Commands chapter of Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference for updates that support Nested Wildcard Apply Policy  
Refer Routing Policy Language Commands chapter for information on the commands used to create, modify, monitor, and maintain routing policies. |
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Introduced/Changed in Release</th>
<th>Where Documented</th>
</tr>
</thead>
</table>
| Editing Routing Policy Language set elements Using XML | This feature was introduced. | Release 4.2.1 | Implementing Routing Policy chapter  
Editing Routing Policy Language set elements Using XML, on page 486 |
| "Set" as a valid operator for the 'med' attribute at the bgp export and bgp import attach points | The BGP Policy Attach Points feature was enhanced to add "Set" as a valid operator for the 'med' attribute at the bgp export and bgp import attach points. | Release 4.2.1 | Implementing Routing Policy chapter  
BGP Policy Attach Points, on page 446  
BGP Attributes and Operators, on page 456 |
| Enhanced Object Tracking for IP Static | This feature was introduced. | Release 4.2.1 | Implementing Static Routes chapter  
Enabling Object Tracking for Static Routes, on page 511 |
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.

For more information about BGP and complete descriptions of the BGP commands listed in this module, see Related Documents, on page 145 section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the Cisco ASR 9000 Series Router software master command index.

Feature History for Implementing BGP

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.7.2</td>
<td>This feature was introduced.</td>
</tr>
<tr>
<td>Release 3.9.0</td>
<td>The following features were supported:</td>
</tr>
<tr>
<td></td>
<td>• BGP Prefix Independent Convergence Unipath Primary Backup</td>
</tr>
<tr>
<td></td>
<td>• BGP Local Label Retention</td>
</tr>
<tr>
<td></td>
<td>• Asplain notation for 4-byte Autonomous System Number</td>
</tr>
<tr>
<td></td>
<td>• BGP Nonstop Routing</td>
</tr>
<tr>
<td></td>
<td>• Command Line Interface (CLI) consistency for BGP commands</td>
</tr>
<tr>
<td></td>
<td>• L2VPN Address Family Configuration Mode</td>
</tr>
<tr>
<td>Release</td>
<td>Modification</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Release 4.0.0</td>
<td>The following features were supported:</td>
</tr>
<tr>
<td></td>
<td>• BGP Add Path Advertisement</td>
</tr>
<tr>
<td></td>
<td>• Accumulated iGP (AiGP)</td>
</tr>
<tr>
<td></td>
<td>• Pre-route</td>
</tr>
<tr>
<td></td>
<td>• IPv4 BGP-Policy Accounting</td>
</tr>
<tr>
<td></td>
<td>• IPv6 uRPF</td>
</tr>
<tr>
<td>Release 4.1.0</td>
<td>Support for 5000 BGP NSR sessions was added</td>
</tr>
<tr>
<td>Release 4.1.1</td>
<td>The following features were added:</td>
</tr>
<tr>
<td></td>
<td>• BGP Accept Own</td>
</tr>
<tr>
<td></td>
<td>• BGP DMZ Link Bandwidth for Unequal Cost Recursive Load Balancing</td>
</tr>
<tr>
<td>Release 4.2.0</td>
<td>The following features were supported:</td>
</tr>
<tr>
<td></td>
<td>• Selective VRF Download</td>
</tr>
<tr>
<td></td>
<td>• BGP Multi-Instance/Multi-AS</td>
</tr>
<tr>
<td></td>
<td>• BFD Multihop Support for BGP</td>
</tr>
<tr>
<td></td>
<td>• BGP Error Handling</td>
</tr>
<tr>
<td></td>
<td>Support for Distributed BGP (bgp distributed speaker) configuration was removed.</td>
</tr>
<tr>
<td>Release 4.2.1</td>
<td>The following features were supported:</td>
</tr>
<tr>
<td></td>
<td>• BGP 3107 PIC Updates for Global Prefixes</td>
</tr>
<tr>
<td></td>
<td>• BGP Prefix Independent Convergence for RIB and FIB</td>
</tr>
<tr>
<td></td>
<td>• BGP Prefix Origin Validation Based on RPKI</td>
</tr>
<tr>
<td>Release 4.2.3</td>
<td>The BGP Attribute Filtering feature was added.</td>
</tr>
<tr>
<td>Release 4.3.1</td>
<td>The following features were supported</td>
</tr>
<tr>
<td></td>
<td>• BGP VRF Dynamic Route Leaking</td>
</tr>
<tr>
<td>Release 4.3.2</td>
<td>The following features were supported</td>
</tr>
<tr>
<td></td>
<td>• Per-neighbor Link Bandwidth</td>
</tr>
</tbody>
</table>
The following features were supported:

- L3VPN iBGP-PE-CE configuration
- Source-based flow tag
- Discard extra paths

The following features were supported:

- Graceful Maintenance
- Per Neighbor TCP MSS

The following features were supported:

- Excessive Punt Flow Trap Processing
- 64-ECMP for BGP

---

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**

For information on configuring BGP to distribute Multiprotocol Label Switching (MPLS) Layer 3 virtual private network (VPN) information, see the *Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide*

For information on BGP support for Bidirectional Forwarding Detection (BFD), see the *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Configuration Guide* and the *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Command Reference*.

---

**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:
  - 512K (524,288) prefixes for IPv4 unicast
  - 128K (131,072) prefixes for IPv6 unicast
  - 128K (131,072) prefixes for IPv4 multicast
  - 512K (524,288) prefixes for VPNv4 unicast
  - IPv4 Unicast: 1048576
  - IPv4 Labeled-unicast: 131072
  - IPv4 Tunnel: 1048576
  - IPv6 Unicast: 524288
  - IPv6 Labeled-unicast: 131072
  - IPv6 Multicast: 131072
  - IPv6 Multicast: 131072
  - VPNv4 Unicast: 2097152
  - IPv4 MDT: 131072
  - VPNv6 Unicast: 1048576
  - L2VPN EVPN: 2097152

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

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

**BGP Next Hop Tracking**

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

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

BGP is notified when any of the following events occurs:

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

---

**Note**

Reachability and 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 `next-hop 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 `next-hop 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 ASR 9000 Series Router module of Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide* (this publication).

**Scoped IPv4/VPNv4 Table Walk**

To determine which address family to process, a next-hop notification is received by first de-referencing the gateway context associated with the next hop, then looking into the gateway context to determine which address families are using the gateway context. The IPv4 unicast and VPNv4 unicast address families share the same gateway context, because they are registered with the IPv4 unicast table in the RIB. As a result, both the global IPv4 unicast table and the VPNv4 table are is processed when an IPv4 unicast next-hop notification is received from the RIB. A mask is maintained in the next hop, indicating if whether the next hop belongs to IPv4 unicast or VPNv4 unicast, or both. This scoped table walk localizes the processing in the appropriate address family table.

**Reordered Address Family Processing**

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

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

**New Thread for Next-Hop Processing**

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

**show, clear, and debug Commands**

The `show bgp nexthops` command provides statistical information about next-hop notifications, the amount of time spent in processing those notifications, and details about each next hop registered with the RIB. The
clear bgp nexthop performance-statistics command ensures that the cumulative statistics associated with the processing part of the next-hop show command can be cleared to help in monitoring. The clear bgp nexthop registration command performs an asynchronous registration of the next hop with the RIB. See the BGP Commands on Cisco ASR 9000 Series Router module of Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference 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 ASR 9000 Series Aggregation Services Router module of Cisco ASR 9000 Series Aggregation Services Router Routing Debug Command Reference.

Autonomous System Number Formats in BGP

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

2-byte Autonomous System Number Format

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

4-byte Autonomous System Number Format

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

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

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

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

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

as-format Command

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

BGP Configuration

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

**Configuration Modes**

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

**Router Configuration Mode**

The following example shows how to enter router configuration mode:

```
RP/0/RSP0/CPU0:router# configuration
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)#
```

**Router Address Family Configuration Mode**

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

```
RP/0/RSP0/CPU0:router(config)# router bgp 112
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)#
```

**Neighbor Configuration Mode**

The following example shows how to enter neighbor configuration mode:

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)#
```

**Neighbor Address Family Configuration Mode**

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

```
RP/0/RSP0/CPU0:router(config)# router bgp 112
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#
```

**VRF Configuration Mode**

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

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_A
RP/0/RSP0/CPU0:router(config-bgp-vrf)#
```
VRF Address Family Configuration Mode

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

```
RP/0/RSP0/CPU0:router(config)# router bgp 112
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_A
RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
```

Configuring Resilient Per-CE Label Allocation Mode Under VRF Address Family

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

**SUMMARY STEPS**

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

**DETAILED STEPS**

**Step 1** configure

Example:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)#
```

Enters global configuration mode.

**Step 2** router bgp as-number

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 666
RP/0/RSP0/CPU0:router(config-bgp)#
```

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

**Step 3** vrf vrf-instance

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf-pe
RP/0/RSP0/CPU0:router(config-bgp-vrf)#
```

Configures a VRF instance.
Step 4  address-family {ipv4 | ipv6} unicast

Example:

RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)#

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

Step 5 label-mode per-ce

Example:

RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# label-mode per-ce
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)#

Configures resilient per-ce label allocation mode.

Step 6 Do one of the following:

- end
- commit

Example:

RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# end
or

RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# commit

Saves configuration changes.

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

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

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

- Use the commit command to save the configuration changes to the running configuration file and remain within the configuration session.

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

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

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

DETAILED STEPS

Step 1 configure  

Example:  
RP/0/RSP0/CPU0:router# configure  
RP/0/RSP0/CPU0:router(config)#  
Enters global configuration mode.

Step 2 route-policy policy-name  

Example:  
RP/0/RSP0/CPU0:router(config)# route-policy route1  
RP/0/RSP0/CPU0:router(config-rpl)#  
Creates a route policy and enters route policy configuration mode.

Step 3 set label-mode per-ce  

Example:  
RP/0/RSP0/CPU0:router(config-rpl)# set label-mode per-ce  
RP/0/RSP0/CPU0:router(config-rpl)#  
Configures resilient per-ce label allocation mode.

Step 4 Do one of the following:  
   • end  
   • commit

Example:  
RP/0/RSP0/CPU0:router(config-rpl)# end  
or  
RP/0/RSP0/CPU0:router(config-rpl)# commit  
Saves configuration changes.
When you issue the **end** command, the system prompts you to commit changes:

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

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

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

---

**VRF Neighbor Configuration Mode**

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

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_A
RP/0/RSP0/CPU0:router(config-bgp-vrf)# neighbor 11.0.1.2
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)#
```

**VRF Neighbor Address Family Configuration Mode**

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

```
RP/0/RSP0/CPU0:router(config)# router bgp 112
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_A
RP/0/RSP0/CPU0:router(config-bgp-vrf)# neighbor 11.0.1.2
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)#
```

**VPNv4 Address Family Configuration Mode**

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

```
RP/0/RSP0/CPU0:router(config)# router bgp 152
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)#
```

**L2VPN Address Family Configuration Mode**

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

```
RP/0/RSP0/CPU0:router(config)# router bgp 100
```
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/RSP0/CPU0:router(config-bgp)# neighbor 192.23.1.2
  RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002
  RP/0/RSP0/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/RSP0/CPU0:router(config-bgp)# neighbor 2002::2
  RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2023
  RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv6 unicast
  RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# next-hop-self
  RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy one in
  ```

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

  ```
  RP/0/RSP0/CPU0:router(config)# router bgp 109
  RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.40.24
  RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1
  RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
  RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# maximum-prefix 1000
  ```

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

  ```
  RP/0/RSP0/CPU0:router(config)# router bgp 110
  RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_A
  RP/0/RSP0/CPU0:router(config-bgp-vrf)# neighbor 11.0.1.2
  RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast
  RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass all in
  ```

Configuration Templates

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

The `af-group` command is used to group address family-specific neighbor commands within an IPv4, IPv6, or VPNv4, 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.
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/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# af-group afmcast1 address-family ipv4 unicast
```

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/RSP0/CPU0:router# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# session-group session1
```

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/RSP0/CPU0:router(config)# router bgp 123
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group nbrgroup1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)#
```

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

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group nbrgroup1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RSP0/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/RSP0/CPU0:router(config)# router bgp 140
   RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group AS_1
   RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
   RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
   RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.1.1.1
   RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1
   RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
   RP/0/RSP0/CPU0:router(config-bgp-nbr)# advertisement-interval 20
   ```

   The following output from the `show bgp neighbors` command shows that the advertisement interval used is 20 seconds:

   ```
   RP/0/RSP0/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/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# session-group AS_2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 20
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use session-group AS_2
RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router(config)# router bgp 150
RP/0/RSP0/CPU0:router(config-bgp)# session-group AS_2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 20
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.1.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router(config-bgp)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# session-group ADV
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group ADV_2
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 25
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# route-policy POLICY_1 in
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.2.2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use session-group ADV
RP/0/RSP0/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/RSP0/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.
```
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/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# remote-as 1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group adv_15
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# remote-as 10
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.101.5
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.101.10
RP/0/RSP0/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/RSP0/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
   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:

```plaintext
RP/0/RSP0/CPU0:router(config)# router bgp 142
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# next-hop-self
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# route-policy POLICY_1 in
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# session-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group GROUP_1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# use session-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# ebgp-multihop 3
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# weight 100
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# send-community-ebgp
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group GROUP_1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# use af-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# weight 200
```

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

```plaintext
RP/0/RSP0/CPU0:router# show bgp neighbors 192.168.0.1 inheritance

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

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

```plaintext
RP/0/RSP0/CPU0:router# show bgp neighbors 192.168.0.1 configuration

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

**show bgp af-group**

Use the `show bgp af-group` command to display address family groups:
- Use the **configuration** keyword to display the effective configuration for the address family group, including any settings that have been inherited from address family groups used by this address family group.

- Use the **inheritance** keyword to display the address family groups from which this address family group is capable of inheriting configuration.

- Use the **users** keyword to display the neighbors, neighbor groups, and address family groups that inherit configuration from this address family group.

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

```plaintext
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afrgrp)# remove-private-as
RP/0/RSP0/CPU0:router(config-bgp-afrgrp)# route-policy POLICY_1 in
RP/0/RSP0/CPU0:router(config-bgp-afrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_1 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# maximum-prefix 2500 75 warning-only
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# default-originate
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# send-community-ebgp
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# send-extended-community-ebgp
RP/0/RSP0/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:

```plaintext
RP/0/RSP0/CPU0:router(config-bgp-afrgrp)# 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:

```plaintext
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# 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:

```plaintext
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# 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/RSP0/CPU0:router(config)# router bgp 113
RP/0/RSP0/CPU0:router(config-bgp)# session-group GROUP_1
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# use session-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# update-source Loopback 0
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# session-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# use session-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# ebgp-multihop 2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# session-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# dmz-link-bandwidth
```

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

```
RP/0/RSP0/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/RSP0/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/RSP0/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:

```plaintext
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# remove-private-as
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# soft-reconfiguration inbound
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# send-community-ebgp
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# send-extended-community-ebgp
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# capability orf prefix both
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# session-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# timers 30 90
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group GROUP_1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# remote-as 1982
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# use neighbor-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# use session-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# use af-group GROUP_2
RP/0/RSP0/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:

```plaintext
RP/0/RSP0/CPU0:router# show bgp neighbor-group GROUP_1 configuration
neighbor-group GROUP_1
remote-as 1982 [n:GROUP_2 s:GROUP_3]
timers 30 90 [n:GROUP_2 s:GROUP_3]
address-family ipv4 unicast [n:GROUP_2 s:GROUP_3]
capability orf prefix-list both [n:GROUP_2 a:GROUP_2]
remove-private-AS [n:GROUP_2 a:GROUP_2]
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]
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/RSP0/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/RSP0/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.

### 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/RSP0/CPU0:router(config)# route-policy pass-all
RP/0/RSP0/CPU0:router(config-rpl)# pass
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
RP/0/RSP0/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 192.168.40.24 and advertise all IPv4 unicast routes back to it:

```
RP/0/RSP0/CPU0# router bgp 1
RP/0/RSP0/CPU0# neighbor 192.168.40.24
RP/0/RSP0/CPU0# remote-as 21
RP/0/RSP0/CPU0# address-family ipv4 unicast
RP/0/RSP0/CPU0# route-policy pass-all in
RP/0/RSP0/CPU0# route-policy pass-all out
RP/0/RSP0/CPU0# 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/RSP0/CPU0# 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 is operating in STANDALONE mode.

<table>
<thead>
<tr>
<th>Process</th>
<th>RecvTblVer</th>
<th>bRIB/RIB</th>
<th>SendTblVer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>41</td>
<td>41</td>
<td>41</td>
</tr>
</tbody>
</table>

Neighbor | Spk | AS | MagRcvd | MsgSent | TblVer | InQ | OutQ | Up/Down | St/PfxRcd |
---------|-----|----|--------|--------|--------|-----|------|---------|----------|
10.0.101.1 | 0   | 1  | 919    | 925    | 41     | 0   | 0    | 0       | 15:15:08 |
10.0.101.2 | 0   | 2  | 0      | 0      | 0      | 0   | 0    | 0       | 00:00:00 |

Address Family: IPv4 Multicast
=================================
BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 1
BGP is operating in STANDALONE mode.

<table>
<thead>
<tr>
<th>Process</th>
<th>RecvTblVer</th>
<th>bRIB/RIB</th>
<th>SendTblVer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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

```
Neighbor | Spk | AS | MagRcvd | MsgSent | TblVer | InQ | OutQ | Up/Down | St/PfxRcd |
---------|-----|----|--------|--------|--------|-----|------|---------|----------|
10.0.101.2 | 0   | 2  | 0      | 0      | 0      | 0   | 0    | 0       | 00:00:00 |

Address Family: IPv6 Unicast
=================================
BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 2
BGP is operating in STANDALONE mode.

<table>
<thead>
<tr>
<th>Process</th>
<th>RecvTblVer</th>
<th>bRIB/RIB</th>
<th>SendTblVer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
```
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 ASR 9000 Series Router` module in the `Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide` for details on table policy use. See the `Cisco Express Forwarding Commands on Cisco ASR 9000 Series Router` module in the `Cisco ASR 9000 Series Aggregation Services Router IP Addresses and Services Command Reference` 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 31
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 ASR 9000 Series Router module of Cisco ASR 9000 Series Aggregation Services Router 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;
- Elself the Cost from one path is less than the other, Choose the path with the lesser Cost as best path;
- Else Continue.

Pathsthatarenotconfiguredwiththecostcommunityattributeareconsideredbythebest-pathselection process to have the default cost value (half of the maximum value [4294967295] or 2147483647).

Applying the cost community attribute at the POI allows you to assign a value to a path originated or learned by a peer in any part of the local autonomous system or confederation. The cost community can be used as a “tie breaker” during the best-path selection process. Multiple instances of the cost community can be configured for separate equal cost paths within the same autonomous system or confederation. For example, a lower cost community value can be applied to a specific exit path in a network with multiple equal cost exit points, and the specific exit path is preferred by the BGP best-path selection process. See the scenario described in Influencing Route Preference in a Multiexit IGP Network, on page 34.

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

See BGP Best Path Algorithm, on page 36 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/RSP0/CPU0:router(config)# route-policy ISP2_PE1
RP/0/RSP0/CPU0:router(config-rpl)# set extcommunity cost (1:1)
```

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

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

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

The “prebest path” point of insertion (POI) in the BGP cost community feature supports mixed EIGRP VPN network topologies that contain VPN and back-door links. This POI is applied automatically to EIGRP routes that are redistributed into BGP. The “prebest path” POI carries the EIGRP route type and metric. This POI influences the best-path calculation process by influencing BGP to consider the POI before any other comparison step. No configuration is required. This feature is enabled automatically for EIGRP VPN sites when Cisco IOS XR software is installed on a PE, CE, or back-door router.
For information about configuring EIGRP MPLS VPNs, see the Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide.

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

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

The following sequence of events happens in PE1:

1. PE1 learns IPv4 prefix 10.1.1.0/24 from CE1 through EIGRP running a virtual routing and forwarding (VRF) instance. EIGRP selects and installs the best path in the RIB. It also encodes the cost-extended community and adds the information to the RIB.

2. The route is redistributed into BGP (assuming that IGP-to-BGP redistribution is configured). BGP also receives the cost-extended community from the route through the redistribution process.

3. After BGP has determined the best path for the newly redistributed prefix, the path is advertised to PE peers (PE2).

4. PE2 receives the BGP VPNv4 prefix route_distinguisher:10.1.1.0/24 along with the cost community. It is likely that CE2 advertises the same prefix (because of the back-door link between CE1 and CE2) to PE2 through EIGRP. PE2 BGP would have already learned the CE route through the redistribution process along with the cost community value.

5. PE2 has two paths within BGP: one with cost community cost1 through multipath BGP (PE1) and another with cost community cost2 through the EIGRP neighbor (CE2).

6. PE2 runs the enhanced BGP best-path calculation.

7. PE2 installs the best path in the RIB passing the appropriate cost community value.

8. PE2 RIB has two paths for 10.1.1.0/24: one with cost community cost2 added by EIGRP and another with the cost community cost1 added by BGP. Because both the route paths have cost community, RIB compares the costs first. The BGP path has the lower cost community, so it is selected and downloaded to the RIB.

9. PE2 RIB redistributes the BGP path into EIGRP with VRF. EIGRP runs a diffusing update algorithm (DUAL) because there are two paths, and selects the BGP-redistributed path.

10. PE2 EIGRP advertises the path to CE2 making the path the next hop for the prefix to send the traffic over the MPLS network.
Adding Routes to the Routing Information Base

If a nonsourced path becomes the best path after the best-path calculation, BGP adds the route to the Routing Information Base (RIB) and passes the cost communities along with the other IGP extended communities. When a route with paths is added to the RIB by a protocol, RIB checks the current best paths for the route and the added paths for cost extended communities. If cost-extended communities are found, the RIB compares the set of cost communities. If the comparison does not result in a tie, the appropriate best path is chosen. If the comparison results in a tie, the RIB proceeds with the remaining steps of the best-path algorithm. If a cost community is not present in either the current best paths or added paths, then the RIB continues with the remaining steps of the best-path algorithm. See BGP Best Path Algorithm, on page 36 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.
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 Comparing Pairs of Paths, on page 36 are used to determine whether MED can be compared between any two paths. Normally, this comparison results in one group for each neighbor AS. If the `bgp bestpath med always` command is configured, then there is just one group containing all the paths.

2  The best path in each group is determined. Determining the best path is achieved by iterating through all paths in the group and keeping track of the best one seen so far. Each path is compared with the best-so-far, and if it is better, it becomes the new best-so-far and is compared with the next path in the group.
3 A set of paths is formed containing the best path selected from each group in Step 2. The overall best path is selected from this set of paths, by iterating through them as in Step 2.

Best Path Change Suppression

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

Note

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

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

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

1 If the existing best path is no longer valid, the change cannot be suppressed.
2 If either the existing or new best paths were received from internal (or confederation) peers or were locally generated (for example, by redistribution), then the change cannot be suppressed. That is, suppression is possible only if both paths were received from external peers.
3 If the paths were received from the same peer (the paths would have the same router-id), the change cannot be suppressed. The router ID is calculated using rules in Comparing Pairs of Paths, on page 36.
4 If the paths have different weights, local preferences, origins, or IGP metrics to their next hops, then the change cannot be suppressed. Note that all these values are calculated using the rules in Comparing Pairs of Paths, on page 36.
5 If the paths have different-length AS paths and the bgp bestpath as-path ignore command is not configured, then the change cannot be suppressed. Again, the AS path length is calculated using the rules in Comparing Pairs of Paths, on page 36.
6 If the MED of the paths can be compared and the MEDs are different, then the change cannot be suppressed. The decision as to whether the MEDs can be compared is exactly the same as the rules in Comparing Pairs of Paths, on page 36, 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 Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference.

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 1: BGP Default Administrative Distances, on page 40.

**Table 1: BGP Default Administrative Distances**

<table>
<thead>
<tr>
<th>Distance</th>
<th>Default Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>External</td>
<td>20</td>
<td>Applied to routes learned from eBGP.</td>
</tr>
<tr>
<td>Internal</td>
<td>200</td>
<td>Applied to routes learned from iBGP.</td>
</tr>
<tr>
<td>Local</td>
<td>200</td>
<td>Applied to routes originated by the router.</td>
</tr>
</tbody>
</table>

Note

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

In most cases, when a route is learned through eBGP, it is installed in the IP routing table because of its distance (20). Sometimes, however, two ASs have an IGP-learned back-door route and an eBGP-learned route. Their policy might be to use the IGP-learned path as the preferred path and to use the eBGP-learned path when the IGP path is down. See Figure 3: Back Door Example, on page 40.

**Figure 3: Back Door Example**

In Figure 3: Back Door Example, on page 40, 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/RSP0/CPU0:router(config)# router bgp 100
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)# network 160.10.0.0/16 backdoor
```

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

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

## Multiprotocol BGP

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

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

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

---

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

---

*Figure 4: Noncongruent Unicast and Multicast Routes, on page 42* illustrates simple unicast and multicast topologies that are incongruent, and therefore are not possible without multiprotocol BGP.

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

**Figure 4: Noncongruent Unicast and Multicast Routes**

![Diagram of noncongruent unicast and multicast routes](image)

**Figure 5: Multicast BGP Environment**, on page 43, is a topology of unicast-only routers and multicast-only routers. The two routers on the left are unicast-only routers (that is, they do not support or are not configured to perform multicast routing). The two routers on the right are multicast-only routers. Routers A and B support both unicast and multicast routing. The unicast-only and multicast-only routers are connected to a single NAP.

In **Figure 5: Multicast BGP Environment**, on page 43, only unicast traffic can travel from Router A to the unicast routers to Router B and back. Multicast traffic could not flow on that path, so another routing table is required. Multicast traffic uses the path from Router A to the multicast routers to Router B and back.

**Figure 5: Multicast BGP Environment**, on page 43 illustrates a multiprotocol BGP environment with a separate unicast route and multicast route from Router A to Router B. Multiprotocol BGP allows these routes to be incongruent. Both of the autonomous systems must be configured for internal multiprotocol BGP (IMBGP) in the figure.
A multicast routing protocol, such as PIM, uses the multicast BGP database to perform Reverse Path Forwarding (RPF) lookups for multicast-capable sources. Thus, packets can be sent and accepted on the multicast topology but not on the unicast topology.

**Figure 5: Multicast BGP Environment**

---

**Route Dampening**

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

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

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

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

---

**Note**

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

---

**BGP Routing Domain Confederation**

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

**BGP Route Reflectors**

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

**Figure 6: Three Fully Meshed iBGP Speakers, on page 45** illustrates a simple iBGP configuration with three iBGP speakers (routers A, B, and C). Without route reflectors, when Router A receives a route from an external neighbor, it must advertise it to both routers B and C. Routers B and C do not readvertise the iBGP learned.
route to other iBGP speakers because the routers do not pass on routes learned from internal neighbors to other internal neighbors, thus preventing a routing information loop.

Figure 6: Three Fully Meshed iBGP Speakers

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

Figure 7: Simple BGP Model with a Route Reflector

The internal peers of the route reflector are divided into two groups: client peers and all other routers in the autonomous system (nonclient peers). A route reflector reflects routes between these two groups. The route reflector and its client peers form a *cluster*. The nonclient peers must be fully meshed with each other, but the
client peers need not be fully meshed. The clients in the cluster do not communicate with iBGP speakers outside their cluster.

**Figure 8: More Complex BGP Route Reflector Model**

![Figure 8: More Complex BGP Route Reflector Model](image)

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

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

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

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

An autonomous system can have multiple route reflectors. A route reflector treats other route reflectors just like other iBGP speakers. A route reflector can be configured to have other route reflectors in a client group or nonclient group. In a simple configuration, the backbone could be divided into many clusters. Each route
reflector would be configured with other route reflectors as nonclient peers (thus, all route reflectors are fully meshed). The clients are configured to maintain iBGP sessions with only the route reflector in their cluster. Usually, a cluster of clients has a single route reflector. In that case, the cluster is identified by the router ID of the route reflector. To increase redundancy and avoid a single point of failure, a cluster might have more than one route reflector. In this case, all route reflectors in the cluster must be configured with the cluster ID so that a route reflector can recognize updates from route reflectors in the same cluster. All route reflectors serving a cluster should be fully meshed and all of them should have identical sets of client and nonclient peers.

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

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

- Originator ID is an optional, nontransitive BGP attribute. It is a 4-byte attribute 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.

**Default Address Family for show Commands**

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

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

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

**MPLS VPN Carrier Supporting Carrier**

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

A backbone carrier offers Border Gateway Protocol and Multiprotocol Label Switching (BGP/MPLS) VPN services. The customer carrier can be either:
An Internet service provider (ISP) (By definition, an ISP does not provide VPN service.)
A BGP/MPLS VPN service provider

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

- BGP takes the place of an Interior Gateway Protocol (IGP) and Label Distribution Protocol (LDP) in a VPN routing and forwarding (VRF) table. You can use BGP to distribute routes and MPLS labels. Using a single protocol instead of two simplifies the configuration and troubleshooting.

- BGP is the preferred routing protocol for connecting two ISPs, mainly because of its routing policies and ability to scale. ISPs commonly use BGP between two providers. This feature enables those ISPs to use BGP.

For detailed information on configuring MPLS VPN CSC with BGP, see the Implementing MPLS Layer 3 VPNs on Cisco ASR 9000 Series Router module of the Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide.

**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 Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide 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 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
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.

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

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

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

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

BGP NSR provides the following features:

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

**BGP Prefix Independent Convergence Unipath Primary/Backup**

The Border Gateway Protocol Prefix Independent Convergence Unipath (BGP PIC Unipath) primary/backup feature provides the capability to install a backup path into the forwarding table. Installing the backup path provides prefix independent convergence in the event of a primary PE–CE link failure.
The primary/backup path provides a mechanism for BGP to determine a backup best path. The backup best path acts as a backup to the overall best path, which is the primary best path. BGP programs both the paths into the Forwarding Information Base (FIB).

The procedure to determine the backup best 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 paths that have the same next hop as that of the current best path.
4. Rerun the best path algorithm on the remaining set of paths to determine the backup best path.

The PE-CE local convergence is in the order of four to five seconds for 10000 prefixes. Installing a backup path on the linecards, so that the Forwarding Information Base (FIB) can immediately switch to an alternate path, in the event of a primary PE-CE link failure reduces the convergence time.

In the case of primary PE-CE link failure, the FIB starts forwarding the received traffic towards the backup PE. FIB will continue forwarding the received traffic towards the backup PE for the duration of the network convergence. Since the approach of using a backup path is independent to the prefixes, Prefix Independent Convergence Unipath functionality provides a prefix independent sub second convergence.

The `additional-paths selection` command installs the backup path in the Forwarding Information Base (FIB) to enable primary backup path.

---

**BGP Local Label Retention**

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

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

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

---

**Command Line Interface (CLI) Consistency for BGP Commands**

From Cisco IOS XR Release 3.9.0 onwards, the Border Gateway Protocol (BGP) commands use `disable` keyword to disable a feature. The keyword `inheritance-disable` disables the inheritance of the feature properties from the parent level.

---

**BGP Additional Paths**

The Border Gateway Protocol (BGP) Additional Paths feature modifies the BGP protocol machinery for a BGP speaker to be able to send multiple paths for a prefix. This gives ‘path diversity’ in the network. The add path enables BGP prefix independent convergence (PIC) at the edge routers.
BGP add path enables add path advertisement in an iBGP network and advertises the following types of paths for a prefix:

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

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

### iBGP Multipath Load Sharing

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

The iBGP Multipath Load Sharing feature enables the BGP speaking router to select multiple iBGP paths as the best paths to a destination. The best paths or multipaths are then installed in the IP routing table of the router.

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

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

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

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

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

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

### Accumulated Interior Gateway Protocol Attribute

The Accumulated Interior Gateway Protocol (AiGP) Attribute is an optional non-transitive BGP Path Attribute. The attribute type code for the AiGP Attribute is to be assigned by IANA. The value field of the AiGP Attribute is defined as a set of Type/Length/Value elements (TLVs). The AiGP TLV contains the Accumulated IGP Metric.
The AiGP feature is required in the 3107 network to simulate the current OSPF behavior of computing the distance associated with a path. OSPF/LDP carries the prefix/label information only in the local area. Then, BGP carries the prefix/label to all the remote areas by redistributing the routes into BGP at area boundaries. The routes/labels are then advertised using LSPs. The next hop for the route is changed at each ABR to local router which removes the need to leak OSPF routes across area boundaries. The bandwidth available on each of the core links is mapped to OSPF cost, hence it is imperative that BGP carries this cost correctly between each of the PEs. This functionality is achieved by using the AiGP.

**Per VRF and Per CE Label for IPv6 Provider Edge**

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

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

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

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

**IPv4 BGP-Policy Accounting on Cisco ASR 9000's A9K-SIP-700**

Border Gateway Protocol (BGP) policy accounting measures and classifies IP traffic that is sent to, or received from, different peers. Policy accounting is enabled on an individual input or output interface basis. Counters based on parameters such as community list, autonomous system number, or autonomous system path are assigned to identify the IP traffic.

Using BGP policy accounting, you can account for traffic according to the route it traverses. Service providers can identify and account for all traffic by customer and bill accordingly.

For more information on BGP policy accounting and how to configure BGP policy accounting, refer the Implementing Cisco Express Forwarding module in Cisco ASR 9000 Series Aggregation Services Router IP Addresses and Services Configuration Guide.

**IPv6 Unicast Routing on Cisco ASR 9000's A9K-SIP-700**

Cisco ASR 9000's A9K-SIP-700 provides complete Internet Protocol Version 6 (IPv6) unicast capability. An IPv6 unicast address is an identifier for a single interface, on a single node. A packet that is sent to a unicast address is delivered to the interface identified by that address. Cisco IOS XR software supports the following IPv6 unicast address types:

- Global aggregatable address
- Site-local address
- Link-local address
- IPv4-compatible IPv6 address
For more information on IPv6 uncase addressing, refer the Implementing Network Stack IPv4 and IPv6 module in Cisco ASR 9000 Series Aggregation Services Router IP Addresses and Services Configuration Guide.

IPv6 uRPF Support on Cisco ASR 9000’s A9K-SIP-700

Unicast IPv6 Reverse Path Forwarding (uRPF) mitigates problems caused by the introduction of malformed or spoofed IP source addresses into a network by discarding IP packets that lack a verifiable IP source address. Unicast RPF does this by doing a reverse lookup in the Cisco Express Forwarding (CEF) table. Therefore, uRPF is possible only if CEF is enabled on the router.

Use the `ipv6 verify unicast source reachable-via {any | rx} [allow-default] [allow-self-ping]` command in interface configuration mode to enable IPv6 uRPF.

For more information on IPv6 uRPF, refer Implementing Cisco Express Forwarding module in Cisco ASR 9000 Series Aggregation Services Router IP Addresses and Services Command Reference.

Constrained Route Distribution for BGP/MPLS Internet Protocol VPNs

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

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

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

Constrained Route Distribution Benefits

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

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

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

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

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

Note

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

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

```
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#router bgp 100
RP/0/RSP0/CPU0:router(config-bgp)#address-family ipv4 rt-filter
RP/0/RSP0/CPU0:router(config-bgp-af)#neighbor 1.1.1.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)#address-family ipv4 rt-filter
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#
```

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

```
RP/0/RSP0/CPU0:router#show bgp ipv4 rt-filter 0:2:0:0/0
BGP routing table entry for 0:2:0:0/0
Versions:
  Process bRIB/RIB  SendTblVer
  8489  8489
Last Modified: Jul 2 10:30:42.452 for 3d03h
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
Local
  0.0.0.0 from 0.0.0.0 (192.192.5.7)
    Received Label 1
    Origin IGP, local pref 100, valid, redistributed, best, group-best
    Received Path ID 0, Local Path ID 1, version 8489
```

Selective VRF Download

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

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

By default, SVD is enabled on the line cards. Use the `selective-vrf-download disable` command to disable SVD. Use the `show svd role` and `show svd state` commands to display role and state information of SVD on line cards.
Line Card Roles and Filters in Selective VRF Download

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

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

The line cards handle these prefixes:

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

These filters are applicable to each line card type:

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

BGP Accept Own

The BGP Accept Own feature enables handling of self-originated VPN routes, which a BGP speaker receives from a route-reflector (RR). A "self-originated" route is one which was originally advertised by the speaker itself. As per BGP protocol [RFC4271], a BGP speaker rejects advertisements that were originated by the speaker itself. However, the BGP Accept Own mechanism enables a router to accept the prefixes it has advertised, when reflected from a route-reflector that modifies certain attributes of the prefix. A special community called ACCEPT-OWN is attached to the prefix by the route-reflector, which is a signal to the receiving router to bypass the ORIGINATOR_ID and NEXTTHOP/MP_REACH_NLRI check. Generally, the BGP speaker detects prefixes that are self-originated through the self-origination check (ORIGINATOR_ID, NEXTTHOP/MP_REACH_NLRI) and drops the received updates. However, with the Accept Own community present in the update, the BGP speaker handles the route.
One of the applications of BGP Accept Own is auto-configuration of extranets within MPLS VPN networks. In an extranet configuration, routes present in one VRF is imported into another VRF on the same PE. Normally, the extranet mechanism requires that either the import-rt or the import policy of the extranet VRFs be modified to control import of the prefixes from another VRF. However, with Accept Own feature, the route-reflector can assert that control without the need for any configuration change on the PE. This way, the Accept Own feature provides a centralized mechanism for administering control of route imports between different VRFs. BGP Accept Own is supported only for VPNv4 and VPNv6 address families in neighbor configuration mode.

Route-Reflector Handling Accept Own Community and RTs

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

Accept Own Configuration Example

In this configuration example:

- CE1 originates prefix X.
- CE1 is configured with Customer VRF and Service VRF.
- OSPF is used as the IGP.
- VPNv4 unicast and VPNv6 unicast address families are enabled between the PE and RR neighbors and IPv4 and IPv6 are enabled between PE and CE neighbors.

The Accept Own configuration works as follows:

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

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

BGP DMZ Link Bandwidth for Unequal Cost Recursive Load Balancing
Border Gateway Protocol demilitarized zone (BGP DMZ) Link Bandwidth for Unequal Cost Recursive Load Balancing provides support for unequal cost load balancing for recursive prefixes on local node using BGP DMZ Link Bandwidth. The unequal load balance is achieved by using the dmz-link-bandwidth command in BGP Neighbor configuration mode and the bandwidth command in Interface configuration mode.

BFD Multihop Support for BGP
Bi-directional Forwarding Detection Multihop (BFD-MH) support is enabled for BGP. BFD Multihop establishes a BFD session between two addresses that may span multiple network hops. Cisco IOS XR Software BFD Multihop is based on RFC 5883. For more information on BFD Multihop, refer Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide and Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Command Reference.

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.

BGP Prefix Origin Validation Based on RPKI

A BGP route associates an address prefix with a set of autonomous systems (AS) that identify the interdomain path the prefix has traversed in the form of BGP announcements. This set is represented as the AS_PATH attribute in BGP and starts with the AS that originated the prefix.

To help reduce well-known threats against BGP including prefix mis-announcing and monkey-in-the-middle attacks, one of the security requirements is the ability to validate the origination AS of BGP routes. The AS number claiming to originate an address prefix (as derived from the AS_PATH attribute of the BGP route) needs to be verified and authorized by the prefix holder.

The Resource Public Key Infrastructure (RPKI) is an approach to build a formally verifiable database of IP addresses and AS numbers as resources. The RPKI is a globally distributed database containing, among other things, information mapping BGP (internet) prefixes to their authorized origin-AS numbers. Routers running BGP can connect to the RPKI to validate the origin-AS of BGP paths.

Configuring RPKI Cache-server

Perform this task to configure Resource Public Key Infrastructure (RPKI) cache-server parameters.

Configure the RPKI cache-server parameters in rpki-server configuration mode. Use the rpki server command in router BGP configuration mode to enter into the rpki-server configuration mode.
SUMMARY STEPS

1. configure
2. router bgp as-number
3. rpki server \{host-name | ip-address\}
4. Use one of these commands:
   - transport ssh port \(port\_number\)
   - transport tcp port \(port\_number\)
5. (Optional) username \(user\_name\)
6. (Optional) password \(password\)
7. preference \(preference\_value\)
8. purge-time \(time\)
9. Use one of these commands.
   - refresh-time \(time\)
   - refresh-time off
10. Use one these commands.
    - response-time \(time\)
    - response-time off
11. commit
12. (Optional) shutdown

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router bgp as-number</td>
<td>Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)#router bgp 100</td>
<td></td>
</tr>
<tr>
<td>Step 3 rpki server {host-name</td>
<td>ip-address}</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-bgp)#rpki server 10.2.3.4</td>
<td></td>
</tr>
<tr>
<td>Step 4 Use one of these commands:</td>
<td>Specifies a transport method for the RPKI cache.</td>
</tr>
<tr>
<td>• transport ssh port (port_number)</td>
<td>• ssh—Select ssh to connect to the RPKI cache using SSH.</td>
</tr>
<tr>
<td>• transport tcp port (port_number)</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command/Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td><code>transport ssh port 1</code> or <code>transport tcp port 2</code></td>
<td>- <strong>tcp</strong>—Select tcp to connect to the RPKI cache using TCP (unencrypted).&lt;br&gt;- <strong>port port_number</strong>—Specify a port number for the specified RPKI cache transport. Range for the port number is 1 to 65535 for both ssh and tcp.</td>
</tr>
<tr>
<td>5</td>
<td><code>username user_name</code></td>
<td>(Optional) Specifies a (SSH) username for the RPKI cache-server.</td>
</tr>
<tr>
<td>6</td>
<td><code>password password</code></td>
<td>(Optional) Specifies a (SSH) password for the RPKI cache-server.</td>
</tr>
<tr>
<td>7</td>
<td><code>preference preference_value</code></td>
<td>Specifies a preference value for the RPKI cache. Range for the preference value is 1 to 10. Setting a lower preference value is better.</td>
</tr>
<tr>
<td>8</td>
<td><code>purge-time time</code></td>
<td>Configures the time BGP waits to keep routes from a cache after the cache session drops. Set purge time in seconds. Range for the purge time is 30 to 360 seconds.</td>
</tr>
<tr>
<td>9</td>
<td>Use one of these commands.</td>
<td>Configures the time BGP waits in between sending periodic serial queries to the cache. Set refresh-time in seconds. Range for the refresh time is 15 to 3600 seconds.&lt;br&gt;Configure the <strong>off</strong> option to specify not to send serial-queries periodically.</td>
</tr>
<tr>
<td>10</td>
<td>Use one of these commands.</td>
<td>Configures the time BGP waits for a response after sending a serial or reset query. Set response-time in seconds. Range for the response time is 15 to 3600 seconds.</td>
</tr>
</tbody>
</table>

Example:

**Step 5**

```
RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#transport ssh port 1
```

**Step 6**

```
RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#username ssh_rpki_uname
```

**Step 7**

```
RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#password ssh_rpki_pass
```

**Step 8**

```
RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#purge-time 30
```

**Step 9**

```
RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#refresh-time 20
```

**Step 10**

```
RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#response-time 20
```
Configuring RPKI Prefix Validation

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

SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>Step 1</td>
<td></td>
</tr>
<tr>
<td>router bgp as-number</td>
<td>Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Step 2</td>
<td></td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)#router bgp 100</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

**Command or Action**

**Step 3**

Use one of these commands.

- `bgp origin-as validation disable`
- `bgp origin-as validation time \{off | prefix_validation_time\}

**Example:**

RP/0/RSP0/CPU0:router(config-bgp)#bgp origin-as validation disable

Or

RP/0/RSP0/CPU0:router(config-bgp)#bgp origin-as validation time 50

Or

RP/0/RSP0/CPU0:router(config-bgp)#bgp origin-as validation time off

Sets the BGP origin-AS validation parameters.

- **disable**—Use `disable` option to disable RPKI origin-AS validation.

- **time**—Use `time` option to either set prefix validation time (in seconds) or to set off the automatic prefix validation after an RPKI update.

  Range for prefix validation time is 5 to 60 seconds.

  Configuring the **disable** option disables prefix validation for all eBGP paths and all eBGP paths are marked as "valid" by default.

  **Note**  The bgp origin-as validation options can also be configured in neighbor and neighbor address family submodes. The neighbor must be an eBGP neighbor. If configured at the neighbor or neighbor address family level, prefix validation disable or time options will be valid only for that specific neighbor or neighbor address family.

**Step 4**

`bgp origin-as validation signal ibgp`

**Example:**

RP/0/RSP0/CPU0:router(config-bgp)#bgp origin-as validation time off

Enables the iBGP signaling of validity state through an extended-community.

This can also be configured in global address family submode.

**Step 5**

`commit`

### Configuring RPKI Bestpath Computation

Perform this task to configure RPKI bestpath computation options.

### SUMMARY STEPS

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

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
</tbody>
</table>
### BGP 3107 PIC Updates for Global Prefixes

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

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

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

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

---

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 2</strong> <strong>router bgp as-number</strong></td>
<td>Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config)#router bgp 100</td>
</tr>
<tr>
<td><strong>Step 3</strong> <strong>bgp bestpath origin-as use validity</strong></td>
<td>Enables the validity states of BGP paths to affect the path's preference in the BGP best path process. This configuration can also be done in router BGP address family submode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)#bgp bestpath origin-as use validity</td>
</tr>
<tr>
<td><strong>Step 4</strong> <strong>bgp bestpath origin-as allow invalid</strong></td>
<td>Allows all &quot;invalid&quot; paths to be considered for BGP bestpath computation. <strong>Note</strong> This configuration can also be done at global address family, neighbor, and neighbor address family submodes. Configuring bgp bestpath origin-as allow invalid in router BGP and address family submodes allow all &quot;invalid&quot; paths to be considered for BGP bestpath computation. By default, all such paths are not bestpath candidates. Configuring bgp bestpath origin-as allow invalid in neighbor and neighbor address family submodes allow all &quot;invalid&quot; paths from that specific neighbor or neighbor address family to be considered as bestpath candidates. The neighbor must be an eBGP neighbor. This configuration takes effect only when the <strong>bgp bestpath origin-as use validity</strong> configuration is enabled.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)#bgp bestpath origin-as allow invalid</td>
</tr>
<tr>
<td><strong>Step 5</strong> <strong>commit</strong></td>
<td></td>
</tr>
</tbody>
</table>
The address-family l2vpn vpls-vpws does not support additional-path. Hence, the l2vpn service that uses address-family l2vpn vpls-vpws does not guarantee PIC convergence time.

The 3107 PIC implementation supports these Cisco IOS XR features:

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

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

**BGP Prefix Independent Convergence for RIB and FIB**

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

The BGP PIC for RIB and FIB feature supports:

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

**BGP Update Message Error Handling**

The BGP UPDATE message error handling changes BGP behavior in handling error UPDATE messages to avoid session reset. Based on the approach described in IETF IDR I-D:draft-ietf-idr-error-handling, the Cisco IOS XR BGP UPDATE Message Error handling implementation classifies BGP update errors into various categories based on factors such as, severity, likelihood of occurrence of UPDATE errors, or type of attributes. Errors encountered in each category are handled according to the draft. Session reset will be avoided as much
as possible during the error handling process. Error handling for some of the categories are controlled by configuration commands to enable or disable the default behavior.

According to the base BGP specification, a BGP speaker that receives an UPDATE message containing a malformed attribute is required to reset the session over which the offending attribute was received. This behavior is undesirable as a session reset would impact not only routes with the offending attribute, but also other valid routes exchanged over the session.

**BGP Attribute Filtering**

The BGP Attribute Filter feature checks integrity of BGP updates in BGP update messages and optimizes reaction when detecting invalid attributes. BGP Update message contains a list of mandatory and optional attributes. These attributes in the update message include MED, LOCAL_PREF, COMMUNITY etc. In some cases, if the attributes are malformed, there is a need to filter these attributes at the receiving end of the router. The BGP Attribute Filter functionality filters the attributes received in the incoming update message. The attribute filter can also be used to filter any attributes that may potentially cause undesirable behavior on the receiving router.

Some of the BGP updates are malformed due to wrong formatting of attributes such as the network layer reachability information (NLRI) or other fields in the update message. These malformed updates, when received, causes undesirable behavior on the receiving routers. Such undesirable behavior may be encountered during update message parsing or during re-advertisement of received NLRIs. In such scenarios, its better to filter these corrupted attributes at the receiving end.

**BGP Attribute Filter Actions**

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

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

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

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

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

**BGP Error Handling and Attribute Filtering Syslog Messages**

When a router receives a malformed update packet, an ios_msg of type ROUTING-BGP-3-MALFORM_UPDATE is printed on the console. This is rate limited to 1 message per minute across all neighbors. For malformed packets that result in actions "Discard Attribute" (A5) or "Local
Repair" (A6), the ios_msg is printed only once per neighbor per action. This is irrespective of the number of malformed updates received since the neighbor last reached an "Established" state.

This is a sample BGP error handling syslog message:

%ROUTING-BGP-3-MALFORM_UPDATE : Malformed UPDATE message received from neighbor 13.0.3.50 - message length 90 bytes,
error flags 0x00000840, action taken "TreatAsWithdraw".
Error details: "Error 0x00000800, Field "Attr-missing", Attribute 1 (Flags 0x00, Length 0), Data []"

This is a sample BGP attribute filtering syslog message for the "discard attribute" action:

[4843.46]RP/0/0/CPU0:Aug 21 17:06:17.919 : bgp[1037]: %ROUTING-BGP-5-UPDATE_FILTERED :
One or more attributes were filtered from UPDATE message received from neighbor 40.0.101.1 - message length 173 bytes,
action taken "DiscardAttr".
Filtering details: "Attribute 16 (Flags 0xc0): Action "DiscardAttr". NLRIs: [IPv4 Unicast] 88.2.0.0/17

This is a sample BGP attribute filtering syslog message for the "treat-as-withdraw" action:

[391.01]RP/0/0/CPU0:Aug 20 19:41:29.243 : bgp[1037]: %ROUTING-BGP-5-UPDATE_FILTERED :
One or more attributes were filtered from UPDATE message received from neighbor 40.0.101.1 - message length 166 bytes,
action taken "TreatAsWdr".
Filtering details: "Attribute 4 (Flags 0xc0): Action "TreatAsWdr". NLRIs: [IPv4 Unicast] 88.2.0.0/17

---

**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.
SUMMARY STEPS

1. configure
2. route-policy route-policy-name
3. end-policy
4. commit
5. configure
6. router bgp as-number
7. bgp router-id ip-address
8. address-family { ipv4 | ipv6 } unicast
9. exit
10. neighbor ip-address
11. remote-as as-number
12. address-family { ipv4 | ipv6 } unicast
13. route-policy route-policy-name { in | out }
14. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config)# route-policy
RP/0/RSP0/CPU0:router(config-rpl)# drop-as-1234
RP/0/RSP0/CPU0:router(config-rpl)# if as-path passes-through '1234' then
RP/0/RSP0/CPU0:router(config-rpl)# apply check-communities
RP/0/RSP0/CPU0:router(config-rpl)# else
RP/0/RSP0/CPU0:router(config-rpl)# pass
RP/0/RSP0/CPU0:router(config-rpl)# endif
```

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

**Example:**

```
RP/0/RSP0/CPU0:router(config)# route-policy
RP/0/RSP0/CPU0:router(config-rpl)# drop-as-1234
RP/0/RSP0/CPU0:router(config-rpl)# if as-path passes-through '1234' then
RP/0/RSP0/CPU0:router(config-rpl)# apply check-communities
RP/0/RSP0/CPU0:router(config-rpl)# else
RP/0/RSP0/CPU0:router(config-rpl)# pass
RP/0/RSP0/CPU0:router(config-rpl)# endif
```

<table>
<thead>
<tr>
<th>Step 3</th>
<th>end-policy</th>
<th>(Optional) Ends the definition of a route policy and exits route policy configuration mode.</th>
</tr>
</thead>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
```

**Step 4** commit

**Step 5** configure

<table>
<thead>
<tr>
<th>Step 6</th>
<th>router bgp as-number</th>
<th>Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</th>
</tr>
</thead>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
```
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>bgp router-id ip-address</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 192.168.70.24</code></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>**address-family { ipv4</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast</code></td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td><strong>exit</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-bgp-af)# exit</code></td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td><strong>neighbor ip-address</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</code></td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td><strong>remote-as as-number</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002</code></td>
</tr>
<tr>
<td><strong>Step 12</strong></td>
<td>**address-family { ipv4</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast</code></td>
</tr>
<tr>
<td><strong>Step 13</strong></td>
<td>**route-policy route-policy-name { in</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy drop-as-1234 in</code></td>
</tr>
<tr>
<td><strong>Step 14</strong></td>
<td><strong>commit</strong></td>
</tr>
</tbody>
</table>

**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. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td>Enters BGP configuration mode for the user specified BGP instance.</td>
</tr>
<tr>
<td>Step 2</td>
<td><code>router bgp as-number [instance instance name]</code></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 100 instance inst1</td>
</tr>
<tr>
<td>Step 3</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

**Configuring a Routing Domain Confederation for BGP**

Perform this task to configure the routing domain confederation for BGP. This includes specifying a confederation identifier and autonomous systems that belong to the confederation.

Configuring a routing domain confederation reduces the internal BGP (iBGP) mesh by dividing an autonomous system into multiple autonomous systems and grouping them into a single confederation. Each autonomous system is fully meshed within itself and has a few connections to another autonomous system in the same confederation. The confederation maintains the next hop and local preference information, and that allows you to retain a single Interior Gateway Protocol (IGP) for all autonomous systems. To the outside world, the confederation looks like a single autonomous system.

**SUMMARY STEPS**

1. configure
2. `router bgp as-number`
3. `bgp confederation identifier as-number`
4. `bgp confederation peers as-number`
5. commit
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router bgp as-number</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router# router bgp 120</td>
</tr>
<tr>
<td>Step 3 bgp confederation identifier as-number</td>
<td>Specifies a BGP confederation identifier.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation identifier 5</td>
</tr>
<tr>
<td>Step 4 bgp confederation peers as-number</td>
<td>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.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1091</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1092</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1093</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1094</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1095</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1096</td>
</tr>
<tr>
<td>Step 5 commit</td>
<td></td>
</tr>
</tbody>
</table>

### 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.

EIGP 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/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#lpts pifib hardware police location 0/2/CPU0
RP/0/RSP0/CPU0:router(config-pifib-policer-per-node)#flow bgp configured rate 4000
RP/0/RSP0/CPU0:router(config-pifib-policer-per-node)#flow bgp known rate 4000
RP/0/RSP0/CPU0:router(config-pifib-policer-per-node)#flow bgp default rate 4000
RP/0/RSP0/CPU0:router(config-pifib-policer-per-node)#commit
```
Logging Neighbor Changes

Logging neighbor changes is enabled by default. Use the `log neighbor changes disable` command to turn off logging. The `no log neighbor changes disable` command can also be used to turn logging back on if it has been disabled.

Adjusting BGP Timers

Perform this task to set the timers for BGP neighbors.

BGP uses certain timers to control periodic activities, such as the sending of keepalive messages and the interval after which a neighbor is assumed to be down if no messages are received from the neighbor during the interval. The values set using the `timers bgp` command in router configuration mode can be overridden on particular neighbors using the `timers` command in the neighbor configuration mode.

### SUMMARY STEPS

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

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><code>router bgp</code> <code>as-number</code></td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 123</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>timers bgp</code> <code>keepalive hold-time</code></td>
<td>Sets a default keepalive time and a default hold time for all neighbors.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# timers bgp 30 90</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>neighbor</code> <code>ip-address</code></td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

Command or Action | Purpose
--- | ---
Step 5 | timers `keepalive hold-time` (Optional) Sets the keepalive timer and the hold-time timer for the BGP neighbor.

**Example:**
```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# timers 60 220
```

Step 6 | commit

### Changing the BGP Default Local Preference Value

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

**SUMMARY STEPS**

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

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>configure</code></td>
</tr>
</tbody>
</table>
| **Step 2** | `router bgp as-number` Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process. **Example:**
```
RP/0/RSP0/CPU0:router(config)# router bgp 120
```
| **Step 3** | `bgp default local-preference value` 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). **Example:**
```
RP/0/RSP0/CPU0:router(config-bgp)# bgp default local-preference 200
```
| **Step 4** | `commit` |
Configuring the MED Metric for BGP

Perform this task to set the multi exit discriminator (MED) to advertise to peers for routes that do not already have a metric set (routes that were received with no MED attribute).

**SUMMARY STEPS**

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

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configure</strong></td>
<td></td>
</tr>
<tr>
<td><code>configure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Configure BGP</strong></td>
<td></td>
</tr>
<tr>
<td><code>router bgp as-number</code></td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
</tr>
<tr>
<td><strong>Set Default Metric</strong></td>
<td></td>
</tr>
<tr>
<td><code>default-metric value</code></td>
<td>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).</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# default metric 10</td>
</tr>
<tr>
<td><strong>Commit</strong></td>
<td></td>
</tr>
<tr>
<td><code>commit</code></td>
<td></td>
</tr>
</tbody>
</table>

**Configuring BGP Weights**

Perform this task to assign a weight to routes received from a neighbor. A weight is a number that you can assign to a path so that you can control the best-path selection process. If you have particular neighbors that you want to prefer for most of your traffic, you can use the `weight` command to assign a higher weight to all routes learned from that neighbor.

**Before You Begin**

- **Note**: The `clear bgp` command must be used for the newly configured weight to take effect.
### SUMMARY STEPS

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

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router bgp <code>as-number</code></td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
</tr>
<tr>
<td><strong>Step 3</strong> neighbor <code>ip-address</code></td>
<td>Creates a neighbor and assigns a remote autonomous system number to it.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</td>
</tr>
<tr>
<td><strong>Step 4</strong> remote-as <code>as-number</code></td>
<td>Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002</td>
</tr>
<tr>
<td><strong>Step 5</strong> address-family `{ ipv4</td>
<td>ipv6 }` unicast</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast</td>
</tr>
<tr>
<td><strong>Step 6</strong> weight <code>weight-value</code></td>
<td>Assigns a weight to all routes learned through the neighbor.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# weight 41150</td>
</tr>
<tr>
<td><strong>Step 7</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>
# Tuning the BGP Best-Path Calculation

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

## SUMMARY STEPS

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

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
</tbody>
</table>
| **Step 2** router bgp  
   `as-number` | Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process. |
| Example:  
   `RP/0/RSP0/CPU0:router(config)# router bgp 126` | |
| **Step 3** 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. |
| Example:  
   `RP/0/RSP0/CPU0:router(config-bgp)# bgp bestpath med missing-as-worst` | |
| **Step 4** 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. |
| Example:  
   `RP/0/RSP0/CPU0:router(config-bgp)# bgp bestpath med always` | |
| **Step 5** bgp bestpath med confed | Enables BGP software to compare MED values for paths learned from confederation peers. |
| Example:  
   `RP/0/RSP0/CPU0:router(config-bgp)# bgp bestpath med confed` | |
### Indicating BGP Back-door Routes

Perform this task to set the administrative distance on an external Border Gateway Protocol (eBGP) route to that of a locally sourced BGP route, causing it to be less preferred than an Interior Gateway Protocol (IGP) route.

#### SUMMARY STEPS

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

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>router bgp as-number</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>address-family { ipv4</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast</td>
</tr>
</tbody>
</table>
Configuring Aggregate Addresses

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

SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router bgp as-number</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family {ipv4</td>
<td>ipv6} unicast</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast</td>
<td>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</td>
</tr>
<tr>
<td><strong>Step 4</strong> aggregate-address address/mask-length [as-set] [as-confed-set] [summary-only] [route-policy route-policy-name]</td>
<td>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.</td>
</tr>
</tbody>
</table>
### Purpose

#### Command or Action

| Step 1 | commit |

#### Summary Steps

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

#### Redistributing iBGP Routes into IGP

Perform this task to redistribute iBGP routes into an Interior Gateway Protocol (IGP), such as Intermediate System-to-Intermediate System (IS-IS) or Open Shortest Path First (OSPF).

### Note

Use of the `bgp redistribute-internal` command requires the `clear route *` command to be issued to reinstall all BGP routes into the IP routing table.

### Caution

Redistributing iBGP routes into IGPs may cause routing loops to form within an autonomous system. Use this command with caution.

### Command or Action

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-af)# aggregate-address 10.0.0.0/8 as-set
```

- 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.
### Redistributing Prefixes into Multiprotocol BGP

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

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

**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 isis `process-id` [ level `{1 | 1-inter-area | 2}` ] [ metric `metric-value` ] [ route-policy `route-policy-name` ]
   - redistribute ospf `process-id` [ match `{external [1 | 2] | internal | nssa-external [1 | 2]}` ] [ metric `metric-value` ] [ route-policy `route-policy-name` ]
   - redistribute ospfv3 `process-id` [ match `{external [1 | 2] | internal | nssa-external [1 | 2]}` ] [ metric `metric-value` ] [ route-policy `route-policy-name` ]
   - redistribute rip [ metric `metric-value` ] [ route-policy `route-policy-name` ]
   - redistribute static [ metric `metric-value` ] [ route-policy `route-policy-name` ]
5. commit
# DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router bgp  as-number</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family { ipv4</td>
<td>ipv6 } unicast</td>
</tr>
<tr>
<td></td>
<td>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast</td>
</tr>
<tr>
<td><strong>Step 4</strong> Do one of the following:</td>
<td>Causes routes from the specified instance to be redistributed into BGP.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>• redistribute connected [ metric metric-value ] [ route-policy route-policy-name ]</td>
<td></td>
</tr>
<tr>
<td>• redistribute eigrp process-id [ match { external</td>
<td>internal }] [ metric metric-value ] [ route-policy route-policy-name ]</td>
</tr>
<tr>
<td>• redistribute isis process-id [ level { 1</td>
<td>1-inter-area</td>
</tr>
<tr>
<td>• redistribute ospf process-id [ match { external [ 1</td>
<td>2 ]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>• redistribute ospfv3 process-id [ match { external [ 1</td>
<td>2 ]</td>
</tr>
<tr>
<td>• redistribute rip [ metric metric-value ] [ route-policy route-policy-name ]</td>
<td></td>
</tr>
<tr>
<td>• redistribute static [ metric metric-value ] [ route-policy route-policy-name ]</td>
<td></td>
</tr>
</tbody>
</table>

Example: RP/0/RSP0/CPU0:router(config-bgp-af)# redistribute ospf 110

<table>
<thead>
<tr>
<th><strong>Step 5</strong> commit</th>
<th></th>
</tr>
</thead>
</table>
Configuring BGP Route Dampening

Perform this task to configure and monitor BGP route dampening.
### SUMMARY STEPS

1. configure
2. router bgp as-number
3. address-family { ipv4 | ipv6 } unicast
4. bgp dampening [ half-life [ reuse suppress max-suppress-time ] | route-policy route-policy-name ]
5. commit
6. show bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } flap-statistics
7. show bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } flap-statistics regexp regular-expression
8. show bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } flap-statistics regexp regular-expression
9. show bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } longer-prefixes
10. clear bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } flap-statistics
11. clear bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } flap-statistics
12. clear bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } flap-statistics regexp regular-expression
13. clear bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } route-policy route-policy-name
14. clear bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } flap-statistics network / mask-length
15. clear bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } flap-statistics ip-address / mask-length
16. show bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } dampened-paths
17. clear bgp { ipv4 { unicast | multicast | labeled-unicast } all | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } [ ipv4 { unicast | labeled-unicast } | ipv6 unicast ] } dampening ip-address / mask-length
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>router bgp as-number</code></td>
<td>Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config)# router bgp 120</code></td>
<td>To see a list of all the possible keywords and arguments for this command, use the CLI help (<code>?</code>).</td>
</tr>
<tr>
<td><strong>Step 3</strong> `address-family { ipv4</td>
<td>ipv6 } unicast`</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast</code></td>
<td>Displays BGP flap statistics.</td>
</tr>
<tr>
<td><strong>Step 4</strong> `bgp dampening [ half-life</td>
<td>reuse suppress max-suppress-time ]</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-bgp-af)# bgp dampening 30 1500 10000 120</code></td>
<td>Displays BGP flap statistics for the specified route policy.</td>
</tr>
<tr>
<td><strong>Step 5</strong> commit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> `show bgp { ipv4 { unicast</td>
<td>multicast</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router# show bgp flap statistics</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> `show bgp { ipv4 { unicast</td>
<td>multicast</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router# show bgp flap-statistics regexp _1$</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> `show bgp { ipv4 { unicast</td>
<td>multicast</td>
</tr>
</tbody>
</table>

---

Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td><strong>RP/0/RSP0/CPU0:router(config)# show bgp flap-statistics route-policy policy_A</strong></td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>**show bgp [ipv4 {unicast</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>RP/0/RSP0/CPU0:router# show bgp flap-statistics 172.20.1.1</strong></td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>**show bgp [ipv4 {unicast</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>RP/0/RSP0/CPU0:router# show bgp flap-statistics 172.20.1.1 longer-prefixes</strong></td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>**clear bgp [ipv4 {unicast</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>RP/0/RSP0/CPU0:router# clear bgp all all flap-statistics</strong></td>
</tr>
<tr>
<td><strong>Step 12</strong></td>
<td>**clear bgp [ipv4 {unicast</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast flap-statistics regexp _1$</strong></td>
</tr>
<tr>
<td><strong>Step 13</strong></td>
<td>**clear bgp [ipv4 {unicast</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast flap-statistics regexp _1$</strong></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast flap-statistics route-policy policy_A</td>
<td></td>
</tr>
</tbody>
</table>

**Step 14**
```
clear bgp [ ipv4 { unicast | multicast | labeled-unicast | all } | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vvpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } | ipv4 { unicast | labeled-unicast } | ipv6 unicast ]] flap-statistics network / mask-length
```
Cleans BGP flap statistics for the specified network.

**Example:** RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast flap-statistics 192.168.40.0/24

**Step 15**
```
clear bgp [ ipv4 { unicast | multicast | labeled-unicast | all } | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vvpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } | ipv4 { unicast | labeled-unicast } | ipv6 unicast ]] flap-statistics ip-address / mask-length
```
Cleans BGP flap statistics for routes received from the specified neighbor.

**Example:** RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast flap-statistics 172.20.1.1

**Step 16**
```
show bgp [ ipv4 { unicast | multicast | labeled-unicast | all } | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vvpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } | ipv4 { unicast | labeled-unicast } | ipv6 unicast ]] dampened-paths
```
Displays the dampened routes, including the time remaining before they are unsuppressed.

**Example:** RP/0/RSP0/CPU0:router# show bgp dampened-paths

**Step 17**
```
clear bgp [ ipv4 { unicast | multicast | labeled-unicast | all } | ipv6 unicast | all { unicast | multicast | all | labeled-unicast } | vvpn4 unicast [ rd rd-address ] | vrf { vrf-name | all } | ipv4 { unicast | labeled-unicast } | ipv6 unicast ]] dampening ip-address / mask-length
```
Clears route dampening information and unsuppresses the suppressed routes.

**Caution** Always use the `clear bgp dampening` command for an individual address-family. The all option for address-families with `clear bgp dampening` should never be used during normal functioning of the system. For example, use `clear bgp ipv4 unicast dampening prefix x.x.x./y`.  

---

Implementing BGP

Configuring BGP Route Dampening
# Applying Policy When Updating the Routing Table

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

## Before You Begin

See the [Implementing Routing Policy on Cisco ASR 9000 Series Router module of Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide](this publication) for a list of the supported attributes and operations that are valid for table policy filtering.

## SUMMARY STEPS

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

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>configure</code></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>router bgp as-number</code></td>
</tr>
</tbody>
</table>

**Example:**
```
RP/0/RSP0/CPU0:router(config)# router bgp 120.6
```

**Step 3** | `address-family { ipv4 | ipv6 } unicast` |

**Example:**
```
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
```

**Step 4** | `table-policy policy-name` |

**Example:**
```
RP/0/RSP0/CPU0:router(config-bgp-af)# table-policy tbl-pcly-A
```

**Step 5** | `commit` |
Setting BGP Administrative Distance

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

**SUMMARY STEPS**

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

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Step 2</td>
<td>router bgp  as-number</td>
<td>Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>address-family  { ipv4</td>
<td>ipv6 }  unicast</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>distance bgp  external-distance internal-distance local-distance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RSP0/CPU0:router(config-bgp-af)# distance bgp 20 20 200</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

**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.

---

**Summary Steps**

1. configure
2. `router bgp as-number`
3. `address-family {ipv4 | ipv6} unicast`
4. `exit`
5. `neighbor-group name`
6. `remote-as as-number`
7. `address-family {ipv4 | ipv6} unicast`
8. `route-policy route-policy-name {in | out}`
9. `exit`
10. `exit`
11. `neighbor ip-address`
12. `use neighbor-group group-name`
13. `remote-as as-number`
14. `commit`

---

**Detailed Steps**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router bgp <em>as-number</em></td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
</tbody>
</table>

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
```
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 3</strong> address-family { ipv4</td>
<td>ipv6 } unicast</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> exit</td>
<td>Exits the current configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-af)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> neighbor-group name</td>
<td>Places the router in neighbor group configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group nbr-grp-A</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> remote-as as-number</td>
<td>Creates a neighbor and assigns a remote autonomous system number to it.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# remote-as 2002</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> address-family { ipv4</td>
<td>ipv6 } unicast</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> route-policy route-policy-name { in</td>
<td>out }</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# route-policy drop-as-1234 in</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> exit</td>
<td>Exits the current configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong> exit</td>
<td>Exits the current configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>11</td>
<td>neighbor \ ip-address</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</td>
</tr>
<tr>
<td>12</td>
<td>use neighbor-group \ group-name</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group nbr-grp-A</td>
</tr>
<tr>
<td>13</td>
<td>remote-as \ as-number</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002</td>
</tr>
<tr>
<td>14</td>
<td>commit</td>
</tr>
</tbody>
</table>

### Configuring a Route Reflector for BGP

Perform this task to configure a route reflector for BGP.

All the neighbors configured with the `route-reflector-client` command are members of the client group, and the remaining iBGP peers are members of the nonclient group for the local route reflector.

Together, a route reflector and its clients form a cluster. A cluster of clients usually has a single route reflector. In such instances, the cluster is identified by the software as the router ID of the route reflector. To increase redundancy and avoid a single point of failure in the network, a cluster can have more than one route reflector. If it does, all route reflectors in the cluster must be configured with the same 4-byte cluster ID so that a route reflector can recognize updates from route reflectors in the same cluster. The `bgp cluster-id` command is used to configure the cluster ID when the cluster has more than one route reflector.

### SUMMARY STEPS

1. configure
2. router bgp \ as-number
3. bgp cluster-id \ cluster-id
4. neighbor \ ip-address
5. remote-as \ as-number
6. address-family \ {ipv4 | ipv6 } unicast
7. route-reflector-client
8. commit
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><code>router bgp as-number</code></td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>bgp cluster-id cluster-id</code></td>
<td>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.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# bgp cluster-id 192.168.70.1</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>neighbor ip-address</code></td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td><code>remote-as as-number</code></td>
<td>Creates a neighbor and assigns a remote autonomous system number to it.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2003</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>`address-family { ipv4</td>
<td>ipv6 } unicast`</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-nbr)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td><code>route-reflector-client</code></td>
<td>Configures the router as a BGP route reflector and configures the neighbor as its client.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-reflector-client</td>
<td></td>
</tr>
<tr>
<td>Step 8</td>
<td><code>commit</code></td>
<td></td>
</tr>
</tbody>
</table>

---

### Configuring BGP Route Filtering by Route Policy

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

See the Implementing Routing Policy on Cisco ASR 9000 Series Router module of Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide (this publication) for a list of the supported attributes and operations that are valid for inbound and outbound neighbor policy filtering.

SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>route-policy name</td>
<td>(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.</td>
</tr>
<tr>
<td>3</td>
<td>end-policy</td>
<td>(Optional) Ends the definition of a route policy and exits route policy configuration mode.</td>
</tr>
<tr>
<td>4</td>
<td>router bgp as-number</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
</tbody>
</table>

**Example:**

**Step 2**

```
RP/0/RSP0/CPU0:router(config)# route-policy
RP/0/RSP0/CPU0:router(config)# drop-as-1234
RP/0/RSP0/CPU0:router(config)# if as-path
   passes-through '1234' then
RP/0/RSP0/CPU0:router(config)# apply
   check-communities
RP/0/RSP0/CPU0:router(config)# else
RP/0/RSP0/CPU0:router(config)# pass
RP/0/RSP0/CPU0:router(config)# endif
```
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router bgp as-number</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
</tbody>
</table>

**Example:**

- **configure**
  - RP/0/RSP0/CPU0:router(config)# configure

- **router bgp as-number**
  - RP/0/RSP0/CPU0:router(config)# router bgp 100

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 5 neighbor ip-address</td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>
  - RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24 |
| Step 6 address-family { ipv4 | ipv6 } unicast | Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. |
| Example: | 
  - RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast |
| Step 7 route-policy route-policy-name { in | out } | Applies the specified policy to inbound routes. |
| Example: | 
  - RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy drop-as-1234 in |
| Step 8 commit |          |
Configuring BGP Next-Hop Trigger Delay

Perform this task to configure BGP next-hop trigger delay. The Routing Information Base (RIB) classifies the dampening notifications based on the severity of the changes. Event notifications are classified as critical and noncritical. This task allows you to specify the minimum batching interval for the critical and noncritical events.

SUMMARY STEPS

1. configure
2. router bgp as-number
3. address-family { ipv4 | ipv6 } unicast
4. nexthop trigger-delay { critical delay | non-critical delay }
5. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>router bgp as-number</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

**Command or Action**

**Step 3**

*address-family { ipv4 | ipv6 } unicast*

**Example:**

RP/0/RSP0/CPU0:router(config-bgp)#

*address-family ipv4 unicast*

**Step 4**

*nexthop trigger-delay { critical delay | non-critical delay }*

**Example:**

RP/0/RSP0/CPU0:router(config-bgp-af)#

*nexthop trigger-delay critical 15000*

**Step 5**

*commit*

### Disabling Next-Hop Processing on BGP Updates

Perform this task to disable next-hop calculation for a neighbor and insert your own address in the next-hop field of BGP updates. Disabling the calculation of the best next hop to use when advertising a route causes all routes to be advertised with the network device as the next hop.

**Note**

Next-hop processing can be disabled for address family group, neighbor group, or neighbor address family.

### SUMMARY STEPS

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

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><code>router bgp as-number</code></td>
<td>Example: <code>RP/0/RSP0/CPU0:router(config)# router bgp 120</code></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td><code>neighbor ip-address</code></td>
<td>Example: <code>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</code></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Creates a neighbor and assigns a remote autonomous system number to it.</td>
</tr>
<tr>
<td><code>remote-as as-number</code></td>
<td>Example: <code>RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 206</code></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>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 (?)).</td>
</tr>
<tr>
<td>`address-family { ipv4</td>
<td>ipv6 } unicast`</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Sets the next-hop attribute for all routes advertised to the specified neighbor to the address of the local router. Disabling the calculation of the best next hop to use when advertising a route causes all routes to be advertised with the local network device as the next hop.</td>
</tr>
<tr>
<td><code>next-hop-self</code></td>
<td>Example: <code>RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# next-hop-self</code></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td></td>
</tr>
<tr>
<td><code>commit</code></td>
<td></td>
</tr>
</tbody>
</table>

### 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.
BGP community and extended-community filtering cannot be configured for iBGP neighbors. Communities and extended-communities are always sent to iBGP neighbors under VPNv4, MDT, IPv4, and IPv6 address families.

SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td>Specified the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Step 2</td>
<td>router bgp as-number</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>neighbor ip-address</td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>remote-as as-number</td>
<td>Creates a neighbor and assigns a remote autonomous system number to it.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>address-family {ipv4 {labeled-unicast</td>
<td>unicast</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv6 unicast</td>
<td>• labeled-unicast</td>
</tr>
</tbody>
</table>
Configuring the BGP Cost Community

Perform this task to configure the BGP cost community.

BGP receives multiple paths to the same destination and it uses the best-path algorithm to decide which is the best path to install in RIB. To enable users to determine an exit point after partial comparison, the cost community is defined to tie-break equal paths during the best-path selection process.
SUMMARY STEPS

1. configure
2. route-policy name
3. set extcommunity cost { cost-extcommunity-set-name | cost-inline-extcommunity-set } [ additive ]
4. end-policy
5. router bgp as-number
6. Do one of the following:
   - default-information originate
   - aggregate-address address/mask-length [ as-set ] [ as-confed-set ] [ summary-only ] [ route-policy route-policy-name ]
   - address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv6 unicast | vpv4 unicast } redistribute connected [ metric metric-value ] [ route-policy route-policy-name ]
   - address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv6 unicast | vpv4 unicast } redistribute eigrp process-id [ match { external | internal } ] [ metric metric-value ] [ route-policy route-policy-name ]
   - address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv6 unicast | vpv4 unicast } redistribute isis process-id [ level { 1 | 1-inter-area | 2 } ] [ metric metric-value ] [ route-policy route-policy-name ]
   - address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv6 unicast | vpv4 unicast } redistribute ospf process-id [ match { external [ 1 | 2 ] | internal | nssa-external [ 1 | 2 ] } ] [ metric metric-value ] [ route-policy route-policy-name ]
   - address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv6 unicast | vpv4 unicast } redistribute ospfv3 process-id [ match { external [ 1 | 2 ] | internal | nssa-external [ 1 | 2 ] } ] [ metric metric-value ] [ route-policy route-policy-name ]
   - address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv6 unicast | vpv4 unicast } redistribute rip [ metric metric-value ] [ route-policy route-policy-name ]
   - address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv6 unicast | vpv4 unicast } redistribute static [ metric metric-value ] [ route-policy route-policy-name ]
   - address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv6 unicast | vpv4 unicast } network { ip-address/prefix-length | ip-address mask } [ route-policy route-policy-name ]
   - neighbor ip-address remote-as as-number address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv4 mdt | ipv6 unicast | ipv6 multicast | vpv4 unicast | vpv6 unicast }
   - route-policy route-policy-name { in | out }
7. Do one of the following:
   • address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv4 mdt | ipv6 unicast | ipv6 multicast | vpv4 unicast | vpv6 unicast } redistribute ospfv3 process-id [ match { external [ 1 | 2 ] | internal | nssa-external [ 1 | 2 ] } ] [ metric metric-value ] [ route-policy route-policy-name ]
   • address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv4 mdt | ipv6 unicast | ipv6 multicast | vpv4 unicast | vpv6 unicast } redistribute rip [ metric metric-value ] [ route-policy route-policy-name ]
   • address-family { ipv4 unicast | ipv4 multicast | ipv4 tunnel | ipv6 unicast | vpv4 unicast } network { ip-address/prefix-length | ip-address mask } [ route-policy route-policy-name ]
8. commit
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>route-policy name</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>set extcommunity cost</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>end-policy</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>router bgp as-number</td>
</tr>
</tbody>
</table>

#### Step 2

```
Example:
RP/0/RSP0/CPU0:router(config)# route-policy costA
```

#### Step 3

```
Example:
RP/0/RSP0/CPU0:router(config)# set extcommunity cost cost_A
```

#### Step 4

```
Example:
RP/0/RSP0/CPU0:router(config)# end-policy
```

#### Step 5

```
Example:
RP/0/RSP0/CPU0:router(config)# router bgp 120
```
### 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 119.
Storing updates from a neighbor works only if either the neighbor is route refresh capable or the `soft-reconfiguration inbound` command is configured. Even if the neighbor is route refresh capable and the `soft-reconfiguration inbound` command is configured, the original routes are not stored unless the `always` option is used with the command. The original routes can be easily retrieved with a route refresh request. Route refresh sends a request to the peer to resend its routing information. The `soft-reconfiguration inbound` command stores all paths received from the peer in an unmodified form and refers to these stored paths during the clear. Soft reconfiguration is memory intensive.

**SUMMARY STEPS**

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

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 <code>configure</code></td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Step 2 <code>router bgp as-number</code></td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
</tr>
<tr>
<td>Step 3 <code>neighbor ip-address</code></td>
<td>Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</td>
</tr>
<tr>
<td>Step 4 `address-family { ipv4</td>
<td>ipv6 } unicast`</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast</td>
</tr>
<tr>
<td>Step 5 <code>soft-reconfiguration inbound [ always]</code></td>
<td>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# soft-reconfiguration inbound always</td>
</tr>
</tbody>
</table>
### Configuring a VPN Routing and Forwarding Instance in BGP

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

See the Software Entitlement on Cisco ASR 9000 Series Router module in *Cisco ASR 9000 Series Aggregation Services Router System Management Configuration Guide* for more information on advanced IP licensing.

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

```
RP/0/RSP0/CPU0:router#LC/0/0/CPU0:Dec 15 17:57:53.653 : rsi_agent[247]:
%LICENSE-ASR9K_LICENSE-2-INFRA_VRF_NEEDED : 5 VRF(s) are configured without license
A9K-iVRF-LIC in violation of the Software Right To Use Agreement.
This feature may be disabled by the system without the appropriate license.
Contact Cisco to purchase the license immediately to avoid potential service interruption.
```

**Note**

An AIP license is not required for configuring L2VPN services.

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

#### Defining Virtual Routing and Forwarding Tables in Provider Edge Routers

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

**SUMMARY STEPS**

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>vrf vrf-name</td>
<td>Configures a VRF instance.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# vrf vrf_pe</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>address-family { ipv4</td>
<td>ipv6 } unicast</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>To see a list of all the possible keywords and arguments for this command, use the CLI help (?).</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>maximum prefix maximum [ threshold ]</td>
<td>Configures a limit to the number of prefixes allowed in a VRF table.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>A maximum number of routes is applicable to dynamic routing protocols as well as static or connected routes.</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-vrf-af)# maximum prefix 2300</td>
<td>You can specify a threshold percentage of the prefix limit using the mid-threshold argument.</td>
</tr>
<tr>
<td>5</td>
<td>import route-policy policy-name</td>
<td>(Optional) Provides finer control over what gets imported into a VRF.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>This import filter discards prefixes that do not match the specified policy-name argument.</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-vrf-af)# import route-policy policy_a</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>import route-target [ as-number : nn</td>
<td>ip-address : nn ]</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target 234:222</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>export route-policy policy-name</td>
<td>(Optional) Provides finer control over what gets exported into a VRF.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>This export filter discards prefixes that do not match the specified policy-name argument.</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-vrf-af)# export route-policy policy_b</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>export route-target [ as-number : nn</td>
<td>ip-address : nn ]</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target 123;234</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>
Configuring the Route Distinguisher

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

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

Perform this task to configure the RD.

SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th></th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>router bgp  as-number</td>
<td>Enters BGP configuration mode allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>bgp router-id  ip-address</td>
<td>Configures a fixed router ID for the BGP-speaking router.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 10.0.0.0</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> vrf vrf-name</td>
<td>Configures a VRF instance.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_pe</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> rd { as-number : nn</td>
<td>Configures the route distinguisher.</td>
<td></td>
</tr>
<tr>
<td>ip-address : nn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>auto</td>
<td>Use the auto keyword if you want the router to automatically assign a unique RD to the VRF.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-vrf)# rd 345:567</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> Do one of the following:</td>
<td>Saves configuration changes.</td>
<td></td>
</tr>
<tr>
<td>• end</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• commit</td>
<td>• When you issue the end command, the system prompts you to commit changes:</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-vrf)# end</td>
<td></td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-vrf)# commit</td>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Entering yes saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC configuration mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Entering no exits the configuration session and returns the router to EXEC configuration mode without committing the configuration changes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Use the commit command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Configuring PE-PE or PE-RR Interior BGP Sessions**

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

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

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

**SUMMARY STEPS**

1. configure
2. router bgp as-number
3. address-family vpnv4 unicast
4. exit
5. neighbor ip-address
6. remote-as as-number
7. description text
8. password { clear | encrypted } password
9. shutdown
10. timers keepalive hold-time
11. update-source type interface-id
12. address-family vpnv4 unicast
13. route-policy route-policy-name in
14. route-policy route-policy-name out
15. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>router bgp as-number</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
```

**Step 3**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>address-family vpnv4 unicast</td>
<td>Enters VPN address family configuration mode.</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
```

**Step 4**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit</td>
<td>Exits the current configuration mode.</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config-bgp-af)# exit
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td><code>neighbor ip-address</code></td>
<td>Configures a PE iBGP neighbor.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.16.1.1</code></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><code>remote-as as-number</code></td>
<td>Assigns the neighbor a remote autonomous system number.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1</code></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><code>description text</code></td>
<td>(Optional) Provides a description of the neighbor. The description is used to save comments and does not affect software function.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-bgp-nbr)# description neighbor 172.16.1.1</code></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>`password { clear</td>
<td>encrypted } password`</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-bgp-nbr)# password encrypted 123abc</code></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td><code>shutdown</code></td>
<td>Terminates any active sessions for the specified neighbor and removes all associated routing information.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-bgp-nbr)# shutdown</code></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td><code>timers keepalive hold-time</code></td>
<td>Set the timers for the BGP neighbor.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-bgp-nbr)# timers 12000 200</code></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td><code>update-source type interface-id</code></td>
<td>Allows iBGP sessions to use the primary IP address from a specific interface as the local address when forming an iBGP session with a neighbor.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source gigabitEthernet 0/1/5/0</code></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td><code>address-family vpnv4 unicast</code></td>
<td>Enters VPN neighbor address family configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast</code></td>
<td></td>
</tr>
</tbody>
</table>
### Configuring Route Reflector to Hold Routes That Have a Defined Set of RT Communities

A provider edge (PE) needs to hold the routes that match the import route targets (RTs) of the VPNs configured on it. The PE router can discard all other VPNv4 routes. But, a route reflector (RR) must retain all VPNv4 routes, because it might peer with PE routers and different PEs might require different RT-tagged VPNv4 (making RRs non-scalable). You can configure an RR to only hold routes that have a defined set of RT communities. Also, a number of the RRs can be configured to service a different set of VPNs (thereby achieving some scalability). A PE is then made to peer with all RRs that service the VRFs configured on the PE. When a new VRF is configured with an RT for which the PE does not already hold routes, the PE issues route refreshes to the RRs and retrieves the relevant VPN routes.

**Note**

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

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

### SUMMARY STEPS

1. configure
2. router bgp  as-number
3. address-family  vpnv4 unicast
4. retain route-target  { all | route-policy  route-policy-name }
5. commit

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><strong>Command or Action</strong></td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------</td>
</tr>
<tr>
<td></td>
<td><code>router bgp as-number</code></td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
```

<table>
<thead>
<tr>
<th>Step 3</th>
<th><strong>Command or Action</strong></th>
<th><strong>Purpose</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>address-family vpv4 unicast</code></td>
<td>Enters VPN address family configuration mode.</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpv4 unicast
```

<table>
<thead>
<tr>
<th>Step 4</th>
<th><strong>Command or Action</strong></th>
<th><strong>Purpose</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>`retain route-target { all</td>
<td>route-policy route-policy-name }`</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config-bgp-af)# retain route-target route-policy rr_ext-comm
```

**Note**  The `all` keyword is not required, because this is the default behavior of a route reflector.

<table>
<thead>
<tr>
<th>Step 5</th>
<th><strong>Command or Action</strong></th>
<th><strong>Purpose</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>commit</code></td>
<td></td>
</tr>
</tbody>
</table>

### Configuring BGP as a PE-CE Protocol

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

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router bgp as-number</td>
<td>Enables BGP routing for a particular VRF on the PE router.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
</tr>
<tr>
<td><strong>Step 3</strong> vrf vrf-name</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_pe_2</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><code>bgp router-id ip-address</code></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-vrf)# bgp router-id 172.16.9.9</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><code>label-allocation-mode per-ce</code></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-vrf)# label-allocation-mode per-ce</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>`address-family { ipv4</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>`network { ip-address / prefix-length</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# network 172.16.5.5/24</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><code>aggregate-address address / mask-length</code></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# aggregate-address 10.0.0.0/24</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 9</strong> exit</td>
<td>Exits the current configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# exit</td>
<td></td>
</tr>
</tbody>
</table>

| **Step 10** neighbor *ip-address* | Configures a CE neighbor. The *ip-address* argument must be a private address. |
| **Example:** RP/0/RSP0/CPU0:router(config-bgp-vrf)# neighbor 10.0.0.0 |

| **Step 11** remote-as *as-number* | Configures the remote AS for the CE neighbor. |
| **Example:** RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# remote-as 2 |

| **Step 12** password { clear | encrypted } password | Enable Message Digest 5 (MD5) authentication on a TCP connection between two BGP neighbors. |
| **Example:** RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# password encrypted 234xyz |

| **Step 13** ebgp-multihop [ *ttl-value* ] | Configures the CE neighbor to accept and attempt BGP connections to external peers residing on networks that are not directly connected. |
| **Example:** RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop 55 |

| **Step 14** | Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. |
| Do one of the following: | To see a list of all the possible keywords and arguments for this command, use the CLI help (?) . |
| • address-family { *ipv4* | *ipv6* } unicast | |
| • address-family { *ipv4* *unicast* | *labeled-unicast* } | |
| **Example:** RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast |

<p>| <strong>Step 15</strong> site-of-origin [ <em>as-number</em> : <em>nn</em> | <em>ip-address</em> : <em>nn</em> ] | Configures the site-of-origin (SoO) extended community. Routes that are learned from this CE neighbor are tagged with the SoO extended community before being advertised to the rest of the PEs. SoO is frequently used to detect loops when as-override is configured on the PE router. If the prefix is looped back to the same site, the PE detects this and does not send the update to the CE. |
| <strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# site-of-origin 234:111 |</p>
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 16 <strong>as-override</strong></td>
<td>Configures AS override on the PE router. This causes the PE router to replace the CE’s ASN with its own (PE) ASN.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# as-override</td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong> This loss of information could lead to routing loops; to avoid loops caused by as-override, use it in conjunction with site-of-origin.</td>
<td></td>
</tr>
<tr>
<td>Step 17 <strong>allowas-in</strong> [ <strong>as-occurrence-number</strong> ]</td>
<td>Allows an AS path with the PE autonomous system number (ASN) a specified number of times. Hub and spoke VPN networks need the looping back of routing information to the HUB PE through the HUB CE. When this happens, due to the presence of the PE ASN, the looped-back information is dropped by the HUB PE. To avoid this, use the allowas-in command to allow prefixes even if they have the PEs ASN up to the specified number of times.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# allowas-in 5</td>
<td></td>
</tr>
<tr>
<td>Step 18 <strong>route-policy</strong> <strong>route-policy-name</strong> <strong>in</strong></td>
<td>Specifies a routing policy for an inbound route. The policy can be used to filter routes or modify route attributes.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pe_ce_in_policy in</td>
<td></td>
</tr>
<tr>
<td>Step 19 <strong>route-policy</strong> <strong>route-policy-name</strong> <strong>out</strong></td>
<td>Specifies a routing policy for an outbound route. The policy can be used to filter routes or modify route attributes.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pe_ce_out_policy out</td>
<td></td>
</tr>
<tr>
<td>Step 20 <strong>commit</strong></td>
<td></td>
</tr>
</tbody>
</table>

### Redistribution of IGPs to BGP

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

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

1. configure
2. router bgp as-number
3. vrf vrf-name
4. address-family { ipv4 | ipv6 } unicast
5. Do one of the following:
   - redistribute connected [ metric metric-value ] [ route-policy route-policy-name ]
   - redistribute eigrp process-id [ match { external | internal } ] [ metric metric-value ] [ route-policy route-policy-name ]
   - redistribute isis process-id [ level { 1 | 1-inter-area | 2 } ] [ metric metric-value ] [ route-policy route-policy-name ]
   - redistribute ospf process-id [ match { external { 1 | 2 } | internal | nssa-external { 1 | 2 } } ] [ metric metric-value ] [ route-policy route-policy-name ]
   - redistribute ospfv3 process-id [ match { external { 1 | 2 } | internal | nssa-external { 1 | 2 } } ] [ metric metric-value ] [ route-policy route-policy-name ]
   - redistribute rip [ metric metric-value ] [ route-policy route-policy-name ]
   - redistribute static [ metric metric-value ] [ route-policy route-policy-name ]
6. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router bgp as-number</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
</tr>
<tr>
<td>Step 3 vrf vrf-name</td>
<td>Enables BGP routing for a particular VRF on the PE router.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_a</td>
</tr>
<tr>
<td>Step 4 address-family { ipv4</td>
<td>ipv6 } unicast</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast</td>
</tr>
</tbody>
</table>
### 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.
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router bgp <em>as-number</em></td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> neighbor <em>ip-address</em></td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> remote-as <em>as-number</em></td>
<td>Creates a neighbor and assigns a remote autonomous system number to it.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> keychain <em>name</em></td>
<td>Configures keychain-based authentication.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-bgp-nbr)# keychain kych_a</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>

### Disabling a BGP Neighbor

Perform this task to administratively shut down a neighbor session without removing the configuration.
SUMMARY STEPS

1. configure
2. router bgp as-number
3. neighbor ip-address
4. shutdown
5. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router bgp as-number</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 127</td>
</tr>
<tr>
<td>Step 3 neighbor ip-address</td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24</td>
</tr>
<tr>
<td>Step 4 shutdown</td>
<td>Disables all active sessions for the specified neighbor.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr)# shutdown</td>
</tr>
<tr>
<td>Step 5 commit</td>
<td></td>
</tr>
</tbody>
</table>

Resetting Neighbors Using BGP Inbound Soft Reset

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

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

1. show bgp neighbors
2. clear bgp { ipv4 { unicast | multicast | all | tunnel } | ipv6 unicast | all { unicast | multicast | all | tunnel } | vpnv4 unicast | vrf { vrf-name | all } { ipv4 unicast | ipv6 unicast } | * | ip-address | as as-number | external } soft [ in [ prefix-filter ] ] out ]

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> show bgp neighbors</td>
<td>Verifies that received route refresh capability from the neighbor is enabled.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router# show bgp neighbors</td>
<td></td>
</tr>
</tbody>
</table>

**Step 2** clear bgp { ipv4 { unicast | multicast | all | tunnel } | ipv6 unicast | all { unicast | multicast | all | tunnel } | vpnv4 unicast | vrf { vrf-name | all } { ipv4 unicast | ipv6 unicast } | * | ip-address | as as-number | external } soft [ in [ prefix-filter ] ] out ]

Example: RP/0/RSP0/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. Reseting neighbors is useful if you change the outbound policy for the neighbors or any other configuration that affects the sending or receiving of routing updates.

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

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

SUMMARY STEPS

1. show bgp neighbors
2. clear bgp { ipv4 { unicast | multicast | all | tunnel } | ipv6 unicast | all { unicast | multicast | all | tunnel } | vpnv4 unicast | vrf { vrf-name | all } { ipv4 unicast | ipv6 unicast } | * | ip-address | as as-number | external } clear bgp { ipv4 | ipv6 | unicast | labeled-unicast } soft [ in [ prefix-filter ] ]
Resetting Neighbors Using BGP Hard Reset

Perform this task to reset neighbors using a hard reset. A hard reset removes the TCP connection to the neighbor, removes all routes received from the neighbor from the BGP table, and then re-establishes the session with the neighbor. If the `graceful` keyword is specified, the routes from the neighbor are not removed from the BGP table immediately, but are marked as stale. After the session is re-established, any stale route that has not been received again from the neighbor is removed.

SUMMARY STEPS

1. clear bgp { ipv4 { unicast | multicast | all | tunnel } | ipv6 unicast | all { unicast | multicast | all | tunnel } | vpnv4 unicast | vrf { vrf-name | all } { ipv4 unicast | ipv6 unicast } | { * | ip-address | as as-number | external } | graceful | soft } { [ in [ prefix-filter ] ] | [ out ] | clear bgp { ipv4 | ipv6 } { unicast | labeled-unicast } }

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> clear bgp { ipv4 { unicast</td>
<td>multicast</td>
</tr>
<tr>
<td><strong>Step 2</strong> clear bgp { ipv4 { unicast</td>
<td>multicast</td>
</tr>
</tbody>
</table>

- 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.
Clearing Caches, Tables, and Databases

Perform this task to remove all contents of a particular cache, table, or database. The `clear bgp` command resets the sessions of the specified group of neighbors (hard reset); it removes the TCP connection to the neighbor, removes all routes received from the neighbor from the BGP table, and then re-establishes the session with the neighbor. Clearing a cache, table, or database can become necessary when the contents of the particular structure have become, or are suspected to be, invalid.

**SUMMARY STEPS**

1. `clear bgp {ipv4 {unicast | multicast | all | tunnel} | ipv6 unicast | all {unicast | multicast | all | tunnel} | vpn4 unicast | vrf {vrf-name | all} {ipv4 unicast | ipv6 unicast} ip-address`
2. `clear bgp external`
3. `clear bgp *`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Clears a specified neighbor.</td>
</tr>
</tbody>
</table>
| `clear bgp {ipv4 {unicast | multicast | all | tunnel} | ipv6 unicast | all {unicast | multicast | all | tunnel} | vpn4 unicast | vrf {vrf-name | all} {ipv4 unicast | ipv6 unicast} ip-address`
| Example: | `RP/0/RSP0/CPU0:router# clear bgp ipv4 172.20.1.1` |
| **Step 2** | Clears all external peers. |
| `clear bgp external`
| Example: | `RP/0/RSP0/CPU0:router# clear bgp external` |
| **Step 3** | Clears all BGP neighbors. |
| `clear bgp *`
| Example: | `RP/0/RSP0/CPU0:router# clear bgp *` |
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> <code>show bgp cidr-only</code></td>
<td>Displays routes with nonnatural network masks (classless interdomain routing [CIDR]) routes.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router# show bgp cidr-only</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>show bgp community community-list [ exact-match ]</code></td>
<td>Displays routes that match the specified BGP community.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router# show bgp community 1081:5 exact-match</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>show bgp regexp regular-expression</code></td>
<td>Displays routes that match the specified autonomous system path regular expression.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router# show bgp regexp &quot;^3 &quot;</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> <code>show bgp</code></td>
<td>Displays entries in the BGP routing table.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router# show bgp</td>
<td></td>
</tr>
</tbody>
</table>
## Displaying BGP Process Information

Perform this task to display specific BGP process information.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 5</strong> show bgp neighbors  ip-address [ advertised-routes</td>
<td>Displays information about the BGP connection to the specified neighbor.</td>
</tr>
<tr>
<td>dampened-routes</td>
<td>- The <strong>advertised-routes</strong> keyword displays all routes the router advertised to the neighbor.</td>
</tr>
<tr>
<td>flap-statistics</td>
<td>- The <strong>dampened-routes</strong> keyword displays the dampened routes that are learned from the neighbor.</td>
</tr>
<tr>
<td>performance-statistics</td>
<td>- The <strong>flap-statistics</strong> keyword displays flap statistics of the routes learned from the neighbor.</td>
</tr>
<tr>
<td>received</td>
<td>- The <strong>performance-statistics</strong> keyword displays performance statistics relating to work done by the BGP process for this neighbor.</td>
</tr>
<tr>
<td>prefix-filter</td>
<td>- The <strong>received</strong> <em>prefix-filter</em> keyword and argument display the received prefix list filter.</td>
</tr>
<tr>
<td>routes ]</td>
<td>- The <strong>routes</strong> keyword displays routes learned from the neighbor.</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router# show bgp neighbors 10.0.101.1
```

| **Step 6** show bgp paths | Displays all BGP paths in the database. |

**Example:**

```
RP/0/RSP0/CPU0:router# show bgp paths
```

| **Step 7** show bgp neighbor-group  group-name configuration | Displays the effective configuration for a specified neighbor group, including any configuration inherited by this neighbor group. |

**Example:**

```
RP/0/RSP0/CPU0:router# show bgp neighbor-group group_1 configuration
```

| **Step 8** show bgp summary | Displays the status of all BGP connections. |

**Example:**

```
RP/0/RSP0/CPU0:router# show bgp summary
```
SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> show bgp process</td>
<td>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.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router# show bgp process</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> show bgp ipv4 unicast summary</td>
<td>Displays a summary of the neighbors for the IPv4 unicast address family.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router# show bgp ipv4 unicast summary</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> show bgp vpnv4 unicast summary</td>
<td>Displays a summary of the neighbors for the VPNv4 unicast address family.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router# show bgp vpnv4 unicast summary</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> show bgp vrf ( vrf-name \mid all )</td>
<td>Displays BGP VPN virtual routing and forwarding (VRF) information.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router# show bgp vrf vrf_A</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> show bgp process detail</td>
<td>Displays detailed process information including the memory used by each of various internal structure types.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router# show bgp processes detail</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> show bgp summary</td>
<td>Displays the status of all BGP connections.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router# show bgp summary</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

**Command or Action**

<table>
<thead>
<tr>
<th>Step 7</th>
<th>show placement program bgp</th>
</tr>
</thead>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router# show placement program bgp
```

**Purpose**

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.

---

<table>
<thead>
<tr>
<th>Step 8</th>
<th>show placement program brib</th>
</tr>
</thead>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router# show placement program brib
```

**Purpose**

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 [ipv4 {unicast | multicast | all} | tunnel] | ipv6 {unicast | all} | all {unicast | multicast | all | tunnel} | vrf {vrf-name | all} [ipv4 unicast | update-group [neighbor ip-address | process-id.index [summary | performance-statistics]]`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| `show bgp [ipv4 {unicast | multicast | all} | tunnel] | ipv6 {unicast | all} | all {unicast | multicast | all | tunnel} | vrf {vrf-name | all} [ipv4 unicast | update-group [neighbor ip-address | process-id.index [summary | performance-statistics]]`

**Purpose**

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.
### Purpose

**Command or Action**  
Example:  
RP/0/RSP0/CPU0:router# show bgp update-group 0.0

**Purpose**  
Process ID range is from 0 to 254. Index range is from 0 to 4294967295.  
- The **summary** keyword displays summary information for neighbors in a particular update group.  
- If no argument is specified, this command displays information for all update groups (for the specified address family).  
- The **performance-statistics** keyword displays performance statistics for an update group.

---

## Configuring BGP Nonstop Routing

### SUMMARY STEPS

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

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**  
configure | Specifies the BGP AS number, and enters the BGP configuration mode, for configuring BGP routing processes. |
| **Step 2**  
**router bgp** *as-number*  
**Example:**  
RP/0/RSP0/CPU0:router(config)# router bgp 120 |  
Activates BGP Nonstop routing.  
**Note**  
BGP supports 5000 NSR sessions. |
| **Step 3**  
**nsr**  
**Example:**  
RP/0/RSP0/CPU0:router(config-bgp)# nsr |  
Activates BGP Nonstop routing.  
**Note**  
BGP supports 5000 NSR sessions. |
| **Step 4**  
**commit** | |
Installing Primary Backup Path for Prefix Independent Convergence (PIC)

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

**SUMMARY STEPS**

1. configure
2. `router bgp as-number`
3. Do one of the following
   - `address-family {vpnv4 unicast | vpnv6 unicast}`
   - `vrf vrf-name {ipv4 unicast | ipv6 unicast}`
4. `additional-paths selection route-policy route-policy-name`
5. `commit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>router bgp as-number</code>&lt;br&gt;&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config)# router bgp 100</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Do one of the following&lt;br&gt;- `address-family {vpnv4 unicast</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><code>additional-paths selection route-policy route-policy-name</code>&lt;br&gt;&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-bgp-af)# additional-paths selection route-policy ap1</td>
</tr>
</tbody>
</table>

**Note**<br>Use the `additional-paths selection` command with an appropriate route-policy to calculate backup paths and to enable Prefix Independent Convergence (PIC) functionality.
The route-policy configuration is a pre-requisite for configuring the additional-paths selection mode for a prefix. This is an example route-policy configuration to use with additional-selection command:
```
route-policy ap1
  set path-selection backup 1 install
end-policy
```

### Step 5

**commit**

---

### Retaining Allocated Local Label for Primary Path

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

#### SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { vpv4 unicast | vpv6 unicast }
4. **retain local-label** *minutes*
5. **commit**

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>configure</strong></td>
<td></td>
</tr>
<tr>
<td><strong>router bgp</strong> <em>as-number</em></td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><strong>address-family</strong> { vpv4 unicast</td>
<td>vpv6 unicast }</td>
</tr>
<tr>
<td><strong>retain local-label</strong> <em>minutes</em></td>
<td>Retains the previously allocated local label for the primary path on the primary PE for 10 minutes after reconvergence.</td>
</tr>
</tbody>
</table>

---

Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x
Configuring BGP Additional Paths

Perform these tasks to configure BGP Additional Paths capability:

**SUMMARY STEPS**

1. configure
2. route-policy route-policy-name
3. if conditional-expression then action-statement else
4. pass endif
5. end-policy
6. router bgp as-number
7. address-family {ipv4 {unicast | multicast} | ipv6 {unicast | multicast | l2vpn vpls-vpws | vpnv4 unicast | vpnv6 unicast}
8. additional-paths receive
9. additional-paths send
10. additional-paths selection route-policy route-policy-name
11. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 route-policy route-policy-name</td>
<td>Defines the route policy and enters route-policy configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router (config)#route-policy add_path_policy</td>
<td></td>
</tr>
<tr>
<td>Step 3 if conditional-expression then action-statement else</td>
<td>Decides the actions and dispositions for the given route.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router (config-rpl)#if community matches-any (*) then set path-selection all advertise else</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 4** | pass endif  
Example:  
RP/0/RSP0/CPU0:router(config-rpl-else)#pass  
RP/0/RSP0/CPU0:router(config-rpl-else)#endif | Passes the route for processing and ends the if statement. |
| **Step 5** | end-policy  
Example:  
RP/0/RSP0/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/RSP0/CPU0:router(config)#router bgp 100 | Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process. |
| **Step 7** | address-family {ipv4 {unicast | multicast} | ipv6 {unicast | multicast | l2vpn vpls-vpws | vpv4 unicast | vpv6 unicast}  
Example:  
RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-bgp-af)#additional-paths selection route-policy add_path_policy | Configures additional paths selection capability for a prefix. |
| **Step 11** | commit | |

### Configuring iBGP Multipath Load Sharing

Perform this task to configure the iBGP Multipath Load Sharing:
SUMMARY STEPS

1. configure
2. router bgp as-number
3. address-family {ipv4|ipv6} {unicast|multicast}
4. maximum-paths ibgp number
5. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router bgp as-number</td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 100</td>
<td></td>
</tr>
<tr>
<td>Step 3 address-family {ipv4</td>
<td>ipv6} {unicast</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 multicast</td>
<td></td>
</tr>
<tr>
<td>Step 4 maximum-paths ibgp number</td>
<td>Configures the maximum number of iBGP paths for load sharing.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-af)# maximum-paths ibgp 30</td>
<td></td>
</tr>
<tr>
<td>Step 5 commit</td>
<td></td>
</tr>
</tbody>
</table>

Originating Prefixes with AiGP

Perform this task to configure origination of routes with the AiGP metric:

Before You Begin

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

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

**SUMMARY STEPS**

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

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>configure</strong></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>route-policy</strong> <code>aigp_policy</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CP00:router(config)# route-policy aigp_policy</code></td>
</tr>
<tr>
<td></td>
<td>Enters route-policy configuration mode and sets the route-policy</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>set aigp-metricigp-cost</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CP00:router(config-rpl)# set aigp-metric igp-cost</code></td>
</tr>
<tr>
<td></td>
<td>Sets the internal routing protocol cost as the aigp metric.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>exit</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CP00:router(config-rpl)# exit</code></td>
</tr>
<tr>
<td></td>
<td>Exits route-policy configuration mode.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>router bgp</strong> <code>as-number</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CP00:router(config)# router bgp 100</code></td>
</tr>
<tr>
<td></td>
<td>Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>address-family</strong> `{ipv4</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CP00:router(config-bgp)# address-family ipv4 unicast</code></td>
</tr>
<tr>
<td></td>
<td>Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>redistribute ospf</strong> <code>osp route-policy plcy_name metric value</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CP00:router(config-bgp-af)#redistribute ospf osp route-policy aigp_policy metric 1</code></td>
</tr>
<tr>
<td></td>
<td>Allows the redistribution of AiBGP metric into OSPF.</td>
</tr>
</tbody>
</table>
Configuring BGP Accept Own

Perform this task to configure BGP Accept Own:

SUMMARY STEPS

1. configure
2. router bgp as-number
3. neighbor ip-address
4. remote-as as-number
5. update-source type interface-path-id
6. address-family {vpnv4 unicast | vpngv6 unicast}
7. accept-own [inheritance-disable]

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
</tr>
<tr>
<td>Step 2</td>
<td>router bgp as-number</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)#router bgp 100</td>
</tr>
<tr>
<td>Step 3</td>
<td>neighbor ip-address</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp)#neighbor 10.1.2.3</td>
</tr>
<tr>
<td>Step 4</td>
<td>remote-as as-number</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr)#remote-as 100</td>
</tr>
<tr>
<td>Step 5</td>
<td>update-source type interface-path-id</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr)#update-source Loopback0</td>
</tr>
</tbody>
</table>
### Enabling BGP Unequal Cost Recursive Load Balancing

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

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

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

#### Note

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

### SUMMARY STEPS

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

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 6** | **address-family** `{vpnv4 unicast | vpng6 unicast}`

**Example:**

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)#address-family vpng6 unicast
```

Specifies the address family as VPNv4 or VPNv6 and enters neighbor address family configuration mode.

**Step 7**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **accept-own** | **[inheritance-disable]**

**Example:**

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#accept-own
```

Enables handling of self-originated VPN routes containing `Accept_Own` community.

Use the `inheritance-disable` keyword to disable the "accept own" configuration and to prevent inheritance of "acceptown" from a parent configuration.
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 2** | router bgp *as-number*  
Example:  
RP/0/RSP0/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/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast  
Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.  
To see a list of all the possible keywords and arguments for this command, use the CLI help (?). |
| **Step 4** | maximum-paths { ebgp | ibgp | eibgp } maximum [ unequal-cost ]  
Example:  
RP/0/RSP0/CPU0:router(config-bgp-af)# maximum-paths ebgp 3  
Configures the maximum number of parallel routes that BGP installs in the routing table.  
**Note**  
- Valid values for maximum-paths are 8 for ASR 9000 Ethernet Line Card and 32 for ASR 9000 Enhanced Ethernet Line Card.  
- The ASR 9000 Ethernet Line Card limits the number of routes to be installed to 8 in the forwarding hardware even though the maximum-path value configured is more than 8.  
  - ebgp maximum: Consider only eBGP paths for multipath.  
  - ibgp maximum [ unequal-cost ]: Consider load balancing between iBGP learned paths.  
  - eibgp maximum: Consider both eBGP and iBGP learned paths for load balancing. eIBGP load balancing always does unequal-cost load balancing.  
When eiBGP is applied, eBGP or iBGP load balancing cannot be configured; however, eBGP and iBGP load balancing can coexist. |
| **Step 5** | exit  
Example:  
RP/0/RSP0/CPU0:router(config-bgp-af)# exit  
Exits the current configuration mode. |
| **Step 6** | neighbor *ip-address*  
Example:  
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.0  
Configures a CE neighbor. The *ip-address* argument must be a private address. |
### Configuration Examples for Implementing BGP

This section provides the following configuration examples:

#### Enabling BGP: Example

The following shows how to enable BGP.

```plaintext
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
```

---

**PurposeCommand or Action**

<table>
<thead>
<tr>
<th>Step 7</th>
<th>dmz-link-bandwidth</th>
<th>Originates a demilitarized-zone (DMZ) link-bandwidth extended community for the link to an eBGP/iBGP neighbor.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 8</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>
bgp router-id 1.1.1.1
address-family ipv4 unicast
aggregate-address 10.2.0.0/24 route-policy set_next_hop_agg_v4
aggregate-address 10.3.0.0/24
redistribute static route-policy set_next_hop_static_v4
address-family ipv4 multicast
aggregate-address 10.2.0.0/24 route-policy set_next_hop_agg_v4
aggregate-address 10.3.0.0/24
redistribute static route-policy set_next_hop_static_v4
address-family ipv6 unicast
aggregate-address 2012::/64 route-policy set_next_hop_agg_v6
aggregate-address 2013::/64
redistribute static route-policy set_next_hop_static_v6
address-family ipv6 multicast
aggregate-address 2012::/64 route-policy set_next_hop_agg_v6
aggregate-address 2013::/64
redistribute static route-policy set_next_hop_static_v6
neighbor 10.0.101.60
remote-as 65000
address-family ipv4 unicast
address-family ipv4 multicast
neighbor 10.0.101.61
remote-as 65000
address-family ipv4 unicast
address-family ipv4 multicast
neighbor 10.0.101.62
remote-as 3
address-family ipv4 unicast
route-policy pass-all in
route-policy pass-all out
address-family ipv4 multicast
route-policy pass-all in
route-policy pass-all out
neighbor 10.0.101.64
remote-as 5
update-source Loopback0
address-family ipv4 unicast
route-policy pass-all in
route-policy pass-all out
address-family ipv4 multicast
route-policy pass-all in
route-policy pass-all out

Displaying BGP Update Groups: Example

The following is sample output from the `show bgp update-group` command run in EXEC configuration mode:

```
show bgp update-group

Update group for IPv4 Unicast, index 0.1:
Attributes:
  Outbound Route map:rm
  Minimum advertisement interval:30
  Messages formatted:2, replicated:2
  Neighbors in this update group:
    10.0.101.92

Update group for IPv4 Unicast, index 0.2:
Attributes:
  Minimum advertisement interval:30
  Messages formatted:2, replicated:2
  Neighbors in this update group:
    10.0.101.91
```
BGP Neighbor Configuration: Example

The following example shows how BGP neighbors on an autonomous system are configured to share information. In the example, a BGP router is assigned to autonomous system 109, and two networks are listed as originating in the autonomous system. Then the addresses of three remote routers (and their autonomous systems) are listed. The router being configured shares information about networks 172.16.0.0 and 192.168.7.0 with the neighbor routers. The first router listed is in a different autonomous system; the second neighbor and remote-as commands specify an internal neighbor (with the same autonomous system number) at address 172.16.200.1; and the third neighbor and remote-as commands specify a neighbor on a different autonomous system.

```plaintext
route-policy pass-all
  pass
end-policy
router bgp 109
  address-family ipv4 unicast
  network 172.16.0.0 255.255.0.0
  network 192.168.7.0 255.255.0.0
  neighbor 172.16.200.1
    remote-as 167
  exit
  address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-all out
    neighbor 172.26.234.2
      remote-as 109
    exit
  address-family ipv4 unicast
    neighbor 172.26.64.19
      remote-as 99
    exit
address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
```

BGP Confederation: Example

The following is a sample configuration that shows several peers in a confederation. The confederation consists of three internal autonomous systems with autonomous system numbers 6001, 6002, and 6003. To the BGP speakers outside the confederation, the confederation looks like a normal autonomous system with autonomous system number 666 (specified using the `bgp confederation identifier` command).

In a BGP speaker in autonomous system 6001, the `bgp confederation peers` command marks the peers from autonomous systems 6002 and 6003 as special eBGP peers. Hence, peers 171.16.232.55 and 171.16.232.56 get the local preference, next hop, and MED unmodified in the updates. The router at 171.19.69.1 is a normal eBGP speaker, and the updates received by it from this peer are just like a normal eBGP update from a peer in autonomous system 666.

```plaintext
router bgp 6001
  bgp confederation identifier 666
  bgp confederation peers
    6002
    6003
  exit
  address-family ipv4 unicast
    neighbor 171.16.232.55
      remote-as 6002
    exit
  address-family ipv4 unicast
```
neighbor 171.16.232.56
remote-as 6003
exit
address-family ipv4 unicast
neighbor 171.19.69.1
remote-as 777

In a BGP speaker in autonomous system 6002, the peers from autonomous systems 6001 and 6003 are configured as special eBGP peers. Peer 171.17.70.1 is a normal iBGP peer, and peer 199.99.99.2 is a normal eBGP peer from autonomous system 700.

router bgp 6002
  bgp confederation identifier 666
  bgp confederation peers
    6001
    6003
  exit
  address-family ipv4 unicast
    neighbor 171.17.70.1
    remote-as 6002
    exit
  address-family ipv4 unicast
    neighbor 171.19.232.57
    remote-as 6001
    exit
  address-family ipv4 unicast
    neighbor 171.19.232.56
    remote-as 6003
    exit
  address-family ipv4 unicast
    neighbor 171.19.99.2
    remote-as 700
    exit
  address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-all out

In a BGP speaker in autonomous system 6003, the peers from autonomous systems 6001 and 6002 are configured as special eBGP peers. Peer 192.168.200.200 is a normal eBGP peer from autonomous system 701.

router bgp 6003
  bgp confederation identifier 666
  bgp confederation peers
    6001
    6002
  exit
  address-family ipv4 unicast
    neighbor 171.19.232.57
    remote-as 6001
    exit
  address-family ipv4 unicast
    neighbor 171.19.232.56
    remote-as 6003
    exit
  address-family ipv4 unicast
    neighbor 192.168.200.200
    remote-as 701
    exit
  address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-all out

The following is a part of the configuration from the BGP speaker 192.168.200.205 from autonomous system 701 in the same example. Neighbor 171.16.232.56 is configured as a normal eBGP speaker from autonomous
system 666. The internal division of the autonomous system into multiple autonomous systems is not known to the peers external to the confederation.

```
router bgp 701
  address-family ipv4 unicast
    neighbor 172.16.232.56
      remote-as 666
    exit
  address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-all out
    exit
  address-family ipv4 unicast
    neighbor 192.168.200.205
      remote-as 701
```

**BGP Route Reflector: Example**

The following example shows how to use an address family to configure internal BGP peer 10.1.1.1 as a route reflector client for both unicast and multicast prefixes:

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

**BGP 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
```
Allocated Local Label Retention: Example

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

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

iBGP Multipath Loadsharing Configuration: Example

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

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

Originating Prefixes With AiGP: Example

The following is a sample configuration for originating prefixes with the AiGP metric attribute:

```bash
route-policy aigp-policy
   set aigp-metric 4
   set aigp-metric igp-cost
end-policy

router bgp 100
   address-family ipv4 unicast
      network 10.2.3.4/24 route-policy aigp-policy
      redistribute ospf osp1 metric 4 route-policy aigp-policy
   *
end
```

BGP Accept Own Configuration: Example

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

```bash
router bgp 100
   neighbor 45.1.1.1
      remote-as 100
      update-source Loopback0
      address-family vpnv4 unicast
      route-policy pass-all in
      accept-own
      route-policy drop_111.x.x.x out
   *
end
```
address-family vpnv6 unicast
  route-policy pass-all in
  accept-own
  route-policy drop_111.x.x.x out
! 
This example shows an InterAS-RR configuration for BGP Accept Own.
router bgp 100
neighbor 45.1.1.1
  remote-as 100
  update-source Loopback0
  address-family vpnv4 unicast
  route-policy rt_stitch1 in
  route-reflector-client
  route-policy add_bgp_ao out
!
  address-family vpnv6 unicast
  route-policy rt_stitch1 in
  route-reflector-client
  route-policy add_bgp_ao out
!
  extcommunity-set rt cs_100:1
  100:1
  end-set
  !
  extcommunity-set rt cs_1001:1
  1001:1
  end-set
  !
  route-policy rt_stitch1
  if extcommunity rt matches-any cs_100:1 then
    set extcommunity rt cs_1000:1 additive
  endif
end-policy
!
route-policy add_bgp_ao
  set community (accept-own) additive
end-policy
!
BGP Unequal Cost Recursive Load Balancing: Example

This is a sample configuration for unequal cost recursive load balancing:

interface Loopback0
  ipv4 address 20.20.20.20 255.255.255.255
  !
interface MgmtEth0/RSP0/CPU0/0
  ipv4 address 8.43.0.10 255.255.255.0
  !
interface TenGigE0/3/0/0
  bandwidth 8000000
  ipv4 address 11.11.11.11 255.255.255.0
  ipv6 address 11:11:0:1::11/64
  !
interface TenGigE0/3/0/1
  bandwidth 7000000
  ipv4 address 11.11.12.11 255.255.255.0
  ipv6 address 11:11:0:2::11/64
  !
interface TenGigE0/3/0/2
  bandwidth 6000000
  ipv4 address 11.11.13.11 255.255.255.0
  ipv6 address 11:11:0:3::11/64
  !
interface TenGigE0/3/0/3
bandwidth 5000000
ipv4 address 11.11.14.11 255.255.255.0
ipv6 address 11:11:0:4::11/64

interface TenGigE0/3/0/4
bandwidth 4000000
ipv4 address 11.11.15.11 255.255.255.0
ipv6 address 11:11:0:5::11/64

interface TenGigE0/3/0/5
bandwidth 3000000
ipv4 address 11.11.16.11 255.255.255.0
ipv6 address 11:11:0:6::11/64

interface TenGigE0/3/0/6
bandwidth 2000000
ipv4 address 11.11.17.11 255.255.255.0
ipv6 address 11:11:0:7::11/64

interface TenGigE0/3/0/7
bandwidth 1000000
ipv4 address 11.11.18.11 255.255.255.0
ipv6 address 11:11:0:8::11/64

description CONNECTED TO IXIA 1/3
transceiver permit pid all

interface TenGigE0/4/0/0
ipv4 address 9.9.9.9 255.255.0.0
ipv6 enable

route-policy pass-all
pass
end-policy

router static
address-family ipv4 unicast
202.153.144.0/24 8.43.0.1

router bgp 100
bgp router-id 20.20.20.20
address-family ipv4 unicast
maximum-paths eibgp 8
redistribute connected

neighbor 11.11.11.12
remote-as 200
dmz-link-bandwidth
address-family ipv4 unicast
route-policy pass-all in
route-policy pass-all out

neighbor 11.11.12.12
remote-as 200
dmz-link-bandwidth
address-family ipv4 unicast
route-policy pass-all in
route-policy pass-all out

neighbor 11.11.13.12
remote-as 200
dmz-link-bandwidth
address-family ipv4 unicast
route-policy pass-all in
route-policy pass-all out

neighbor 11.11.14.12
Where to Go Next

For detailed information about BGP commands, see Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference.

Additional References

The following sections provide references related to implementing BGP.

Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td>Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference</td>
</tr>
</tbody>
</table>
### Related Topic

<table>
<thead>
<tr>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco ASR 9000 Series Aggregation Services Router IP Addresses and Services Command Reference</td>
</tr>
<tr>
<td>Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide</td>
</tr>
<tr>
<td>Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide and Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Command Reference</td>
</tr>
<tr>
<td>Configuring AAA Services on Cisco ASR 9000 Series Router module of Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide</td>
</tr>
</tbody>
</table>

### Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>draft-ietf-idr-bgp4-26.txt</td>
<td>A Border Gateway Protocol 4, by Y. Rekhter, T.Li, S. Hares</td>
</tr>
<tr>
<td>draft-ietf-idr-bgp4-mib-15.txt</td>
<td>Definitions of Managed Objects for the Fourth Version of Border Gateway Protocol (BGP-4), by J. Hass and S. Hares</td>
</tr>
<tr>
<td>draft-ietf-idr-cease-subcode-05.txt</td>
<td>Subcodes for BGP Cease Notification Message, by Enke Chen, V. Gillet</td>
</tr>
<tr>
<td>draft-ietf-idr-avoid-transition-00.txt</td>
<td>Avoid BGP Best Path Transitions from One External to Another, by Enke Chen, Srihari Sangli</td>
</tr>
<tr>
<td>draft-ietf-idr-as4bytes-12.txt</td>
<td>BGP Support for Four-octet AS Number Space, by Quaizar Vohra, Enke Chen</td>
</tr>
</tbody>
</table>
### MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>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: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></td>
</tr>
</tbody>
</table>

### RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 1700</td>
<td>Assigned Numbers</td>
</tr>
<tr>
<td>RFC 1997</td>
<td>BGP Communities Attribute</td>
</tr>
<tr>
<td>RFC 2385</td>
<td>Protection of BGP Sessions via the TCP MD5 Signature Option</td>
</tr>
<tr>
<td>RFC 2439</td>
<td>BGP Route Flap Damping</td>
</tr>
<tr>
<td>RFC 2545</td>
<td>Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing</td>
</tr>
<tr>
<td>RFC 2796</td>
<td>BGP Route Reflection - An Alternative to Full Mesh IBGP</td>
</tr>
<tr>
<td>RFC 2858</td>
<td>Multiprotocol Extensions for BGP-4</td>
</tr>
<tr>
<td>RFC 2918</td>
<td>Route Refresh Capability for BGP-4</td>
</tr>
<tr>
<td>RFC 3065</td>
<td>Autonomous System Confederations for BGP</td>
</tr>
<tr>
<td>RFC 3392</td>
<td>Capabilities Advertisement with BGP-4</td>
</tr>
<tr>
<td>RFC 4271</td>
<td>A Border Gateway Protocol 4 (BGP-4)</td>
</tr>
<tr>
<td>RFC 4364</td>
<td>BGP/MPLS IP Virtual Private Networks (VPNs)</td>
</tr>
<tr>
<td>RFC 4724</td>
<td><em>Graceful Restart Mechanism for BGP</em></td>
</tr>
</tbody>
</table>
### Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>able technical content, including links to products, technologies,</td>
<td></td>
</tr>
<tr>
<td>solutions, technical tips, and tools. Registered Cisco.com users</td>
<td></td>
</tr>
<tr>
<td>can log in from this page to access even more content.</td>
<td></td>
</tr>
</tbody>
</table>
Implementing BFD

This module describes the configuration of bidirectional forwarding detection (BFD) on the Cisco ASR 9000 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

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.7.2</td>
<td>BFD was introduced.</td>
</tr>
</tbody>
</table>
| Release 3.9.0 | • Support for these applications with BFD was added:  
  ◦ Hot Standby Router Protocol (HSRP)  
  ◦ Virtual Router Redundancy Protocol (VRRP)  
  ◦ The dampening command was added to minimize BFD session flapping and delay session startup.  
  ◦ The echo ipv4 source command was added to specify a source IP address and override the default.  
  ◦ The ipv6 checksum command was added to enable and disable the IPv6 UDP checksum computation and BFD interface configuration modes. |
<table>
<thead>
<tr>
<th>Release</th>
<th>Support for these BFD features was added:</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0.0</td>
<td>- BFD for OSPFv3</td>
</tr>
<tr>
<td></td>
<td>- BFD for IPv6</td>
</tr>
<tr>
<td></td>
<td>Support for BFD was added on the following SPAs:</td>
</tr>
<tr>
<td></td>
<td>- 1-Port OC-192c/STM-64 POS/RPR XFP SPA</td>
</tr>
<tr>
<td></td>
<td>- 2-Port OC-48c/STM-16 POS/RPR SPA</td>
</tr>
<tr>
<td></td>
<td>- 8-Port OC-12c/STM-4 POS SPA</td>
</tr>
<tr>
<td>4.0.1</td>
<td>- Support for BFD Per Member Links on Link Bundles was added.</td>
</tr>
<tr>
<td></td>
<td>- The <code>echo latency detect</code> command was added to enable latency detection for BFD echo packets on non-bundle interfaces.</td>
</tr>
<tr>
<td></td>
<td>- The <code>echo startup validate</code> command was added to verify the echo path before starting a BFD session on non-bundle interfaces.</td>
</tr>
<tr>
<td>4.2.0</td>
<td>- BFD Multihop Global TTL check.</td>
</tr>
<tr>
<td></td>
<td>- BFD Multihop support for BGP and IPv4 traffic.</td>
</tr>
<tr>
<td></td>
<td>- The <code>multihop ttl-drop-threshold</code> command was added to specify the TTL value to start dropping packets for multihop sessions.</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Support for BFD Multihop feature was added on the ASR9K-SIP-700 line card.</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Support for BFD over Bundles CISCO/IETF mode support on a per bundle basis was added.</td>
</tr>
</tbody>
</table>

- Prerequisites for Implementing BFD, page 151
- Restrictions for Implementing BFD, page 151
- Information About BFD, page 152
- How to Configure BFD, page 166
- Configuration Examples for Configuring BFD, page 189
- Where to Go Next, page 195
- Additional References, page 195
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.

- On the Cisco ASR 9000 Series Router, each line card supporting BFD must be able to perform the following tasks:
  - Send echo packets every 50ms * 3 (as a minimum under normal conditions)
  - Send control packets every 150ms * 3 (as a minimum under stress conditions)
  - Send and receive up to 9600 User Datagram Protocol (UDP) pps. This sustains 144 sessions at a 15-ms echo interval (or 1440 sessions at a 150-ms echo interval).

- To enable BFD for a neighbor, the neighbor router must support BFD.

- In Cisco IOS XR releases before Release 3.9.0, we recommended that you configure the local router ID with the `router-id` command in global configuration mode prior to setting up a BFD session. If you did not configure the local router ID, then by default the source address of the IP packet for BFD echo mode is the IP address of the output interface. Beginning in Cisco IOS XR release 3.9.0 and later, you can use the `echo ipv4 source` command to specify the IP address that you want to use as the source address.

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

Restrictions for Implementing BFD

These restrictions apply to BFD:

- Demand mode is not supported in Cisco IOS XR software.
- BFD echo mode is not supported for these features.
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. For more information, see the Enabling BFD on a Static Route.

- 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 Multipath Sessions Support on nV Edge System**

The following BFD Multipath Sessions are supported on nV Edge System:

- BFD over GRE
- BFD over Logical Bundle
- BFD over IRB
- BFD Multihop (only supported from 5.2.2 onwards)
- BFD over MPLS TE
- BFD over Satellite

**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 9: 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 10: BFD Asynchronous Mode With Echo Packets

For more information about control and echo packet intervals in asynchronous mode, see the BFD Packet Intervals and Failure Detection.

BFD Packet Information

BFD Source and Destination Ports

BFD payload control packets are encapsulated in UDP packets, using destination port 3784 and source port 49152. Even on shared media, like Ethernet, BFD control packets are always sent as unicast packets to the BFD peer.

Echo packets are encapsulated in UDP packets, as well, using destination port 3785 and source port 3785. The BFD over bundle member feature increments each byte of the UDP source port on echo packets with each transmission. UDP source port ranges from 0xC0C0 to 0xFFFF. For example:

1st echo packet: 0xC0C0
2nd echo packet: 0xC1C1
3rd echo packet: 0xC2C2

The UDP source port is incremented so that sequential echo packets are hashed to deviating bundle member.

BFD Packet Intervals and Failure Detection

BFD uses configurable intervals and multipliers to specify the periods at which control and echo packets are sent in asynchronous mode and their corresponding failure detection.

There are differences in how these intervals and failure detection times are implemented for BFD sessions running over physical interfaces, and BFD sessions on bundle member links.

BFD Packet Intervals on Physical Interfaces

When BFD is running over physical interfaces, echo mode is used only if the configured interval is less than two seconds.
BFD sessions running over physical interfaces when echo mode is enabled send BFD control packets at a slow rate of every two seconds. There is no need to duplicate control packet failure detection at a fast rate because BFD echo packets are already being sent at fast rates and link failures will be detected when echo packets are not received within the echo failure detection time.

**BFD Packet Intervals on Bundle Member Links**

On each bundle member interface, BFD asynchronous mode control packets run at user-configurable interval and multiplier values, even when echo mode is running.

However, on a bundle member interface when echo mode is enabled, BFD asynchronous mode must continue to run at a fast rate because one of the requirements of enabling BFD echo mode is that the bundle member interface is available in BFD asynchronous mode.

The maximum echo packet interval for BFD on bundle member links is the minimum of either 30 seconds or the asynchronous control packet failure detection time.

When echo mode is disabled, the behavior is the same as BFD over physical interfaces, where sessions exchange BFD control packets at the configured rate.

**Control Packet Failure Detection In Asynchronous Mode**

Control packet failure in asynchronous mode without echo is detected using the values of the minimum interval (bfd minimum-interval for non-bundle interfaces, and bfd address-family ipv4 minimum-interval for bundle interfaces) and multiplier (bfd multiplier for non-bundle interfaces, and bfd address-family ipv4 multiplier for bundle interfaces) commands.

For control packet failure detection, the local multiplier value is sent to the neighbor. A failure detection timer is started based on \((I \times M)\), where \(I\) is the negotiated interval, and \(M\) is the multiplier provided by the remote end.

Whenever a valid control packet is received from the neighbor, the failure detection timer is reset. If a valid control packet is not received from the neighbor within the time period \((I \times M)\), then the failure detection timer is triggered, and the neighbor is declared down.

**Echo Packet Failure Detection In Asynchronous Mode**

The standard echo failure detection scheme is done through a counter that is based on the value of the `bfd multiplier` command on non-bundle interfaces, and the value of the `bfd address-family ipv4 multiplier` command for bundle interfaces.

This counter is incremented each time the system sends an echo packet, and is reset to zero whenever any echo packet is received, regardless of the order that the packet was sent in the echo packet stream.

Under ideal conditions, this means that BFD generally detects echo failures that exceed the period of time \((I \times M)\) or \((I \times M \times M)\) for bundle interfaces, where:

- \(I\) — Value of the minimum interval (bfd minimum-interval for non-bundle interfaces, and `bfd address-family ipv4 minimum-interval` for bundle interfaces).
- \(M\) — Value of the multiplier (`bfd multiplier` for non-bundle interfaces, and `bfd address-family ipv4 multiplier` for bundle interfaces) commands.

So, if the system transmits one additional echo packet beyond the multiplier count without receipt of any echo packets, echo failure is detected and the neighbor is declared down (See Example 2).
However, this standard echo failure detection does not address latency between transmission and receipt of any specific echo packet, which can build beyond \((I \times M)\) over the course of the BFD session. In this case, BFD will not declare a neighbor down as long as any echo packet continues to be received within the multiplier window and resets the counter to zero. Beginning in Cisco IOS XR 4.0.1, you can configure BFD to measure this latency for non-bundle interfaces. For more information, see Example 3 and the Echo Packet Latency.

**Echo Failure Detection Examples**

This section provides examples of several scenarios of standard echo packet processing and failure detection without configuration of latency detection for non-bundle interfaces. In these examples, consider an interval of 50 ms and a multiplier of 3.

---

**Example 1**

The following example shows an ideal case where each echo packet is returned before the next echo is transmitted. In this case, the counter increments to 1 and is returned to 0 before the next echo is sent and no echo failure occurs. As long as the roundtrip delay for echo packets in the session is less than the minimum interval, this scenario occurs:

<table>
<thead>
<tr>
<th>Time (T)</th>
<th>Echo#1 TX (count = 1)</th>
<th>Echo#2 RX (count = 0)</th>
<th>Echo#3 TX (count = 1)</th>
<th>Echo#4 RX (count = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T + 50 ms</td>
<td>Echo#2 TX (count = 1)</td>
<td>Echo#3 RX (count = 0)</td>
<td>Echo#4 TX (count = 1)</td>
<td>Echo#4 RX (count = 0)</td>
</tr>
<tr>
<td>T + 100 ms</td>
<td>Echo#3 TX (count = 1)</td>
<td>Echo#4 RX (count = 0)</td>
<td>Echo#4 TX (count = 1)</td>
<td>Echo#4 RX (count = 0)</td>
</tr>
</tbody>
</table>

**Example 2**

The following example shows the absence in return of any echo packets. After the transmission of the fourth echo packet, the counter exceeds the multiplier value of 3 and echo failure is detected. In this case, echo failure detection occurs at the 150 ms \((I \times M)\) window:

<table>
<thead>
<tr>
<th>Time (T)</th>
<th>Echo#1 TX (count = 1)</th>
<th>Echo#2 TX (count = 2)</th>
<th>Echo#3 TX (count = 3)</th>
<th>Echo#4 TX (count = 4 - echo failure)</th>
</tr>
</thead>
</table>

**Example 3**

The following example shows an example of how roundtrip latency can build beyond \((I \times M)\) for any particular echo packet over the course of a BFD session using the standard echo failure detection, but latency between return of echo packets overall in the session never exceeds the \((I \times M)\) window and the counter never exceeds the multiplier, so the neighbor is not declared down.
You can configure BFD to detect roundtrip latency on non-bundle interfaces using the `echo latency detect` command beginning in Cisco IOS XR 4.0.1.

<table>
<thead>
<tr>
<th>Time (T)</th>
<th>Event</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>T + 1 ms</td>
<td>Echo#1 TX</td>
<td>1</td>
</tr>
<tr>
<td>T + 5 ms</td>
<td>Echo#1 RX</td>
<td>0</td>
</tr>
<tr>
<td>T + 51 ms</td>
<td>Echo#2 TX</td>
<td>1</td>
</tr>
<tr>
<td>T + 100 ms</td>
<td>Echo#3 TX</td>
<td>1</td>
</tr>
<tr>
<td>T + 150 ms</td>
<td>Echo#4 TX</td>
<td>2</td>
</tr>
<tr>
<td>T + 200 ms</td>
<td>Echo#5 TX</td>
<td>1</td>
</tr>
<tr>
<td>T + 250 ms</td>
<td>Echo#6 TX</td>
<td>2</td>
</tr>
<tr>
<td>T + 300 ms</td>
<td>Echo#7 TX</td>
<td>1</td>
</tr>
<tr>
<td>T + 350 ms</td>
<td>Echo#8 TX</td>
<td>2</td>
</tr>
<tr>
<td>T + 351 ms</td>
<td>Echo#9 RX</td>
<td>0</td>
</tr>
<tr>
<td>T + 400 ms</td>
<td>Echo#10 RX</td>
<td>0</td>
</tr>
</tbody>
</table>

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: 100 ms
- Echo#3 RX - Echo#4 RX: 100 ms
- Echo#4 RX - Echo#5 RX: 100 ms
- Echo#5 RX - Echo#6 RX: 100 ms
- Echo#6 RX - Echo#7 RX: 50 ms
- Echo#7 RX - Echo#8 RX: 50 ms

### 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 2: 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 2: BFD Packet Intervals and Failure Detection Time Examples on Bundle Interfaces

<table>
<thead>
<tr>
<th>Configured Async Control Packet Interval (ms) (bfd address-family ipv4 minimum-interval)</th>
<th>Configured Multiplier (bfd address-family ipv4 multiplier)</th>
<th>Async Control Packet Failure Detection Time (ms) (Interval x Multiplier)</th>
<th>Echo Packet Interval (Async Control Packet Failure Detection Time)</th>
<th>Echo Packet Failure Detection Time (Echo Interval x Multiplier)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>3</td>
<td>150</td>
<td>150</td>
<td>450</td>
</tr>
<tr>
<td>75</td>
<td>4</td>
<td>300</td>
<td>300</td>
<td>1200</td>
</tr>
<tr>
<td>200</td>
<td>2</td>
<td>400</td>
<td>400</td>
<td>800</td>
</tr>
<tr>
<td>2000</td>
<td>3</td>
<td>6000</td>
<td>6000</td>
<td>18000</td>
</tr>
<tr>
<td>15000</td>
<td>3</td>
<td>45000</td>
<td>30000</td>
<td>90000</td>
</tr>
</tbody>
</table>

1 The maximum echo packet interval for BFD on bundle member links is the minimum of either 30 seconds or the asynchronous control packet failure detection time.

Echo Packet Latency

In Cisco IOS XR software releases prior to Cisco IOS XR 4.0.1, BFD only detects an absence of receipt of echo packets, not a specific delay for TX/RX of a particular echo packet. In some cases, receipt of BFD echo packets in general can be within their overall tolerances for failure detection and packet transmission, but a longer delay might develop over a period of time for any particular roundtrip of an echo packet (See Example 3).

Beginning in Cisco IOS XR Release 4.0.1, you can configure the router to detect the actual latency between transmitted and received echo packets on non-bundle interfaces and also take down the session when the latency exceeds configured thresholds for that roundtrip latency. For more information, see the Configuring BFD Session Teardown Based on Echo Latency Detection.

In addition, you can verify that the echo packet path is within specified latency tolerances before starting a BFD session. With echo startup validation, an echo packet is periodically transmitted on the link while it is down to verify successful transmission within the configured latency before allowing the BFD session to change state. For more information, see the Delaying BFD Session Startup Until Verification of Echo Path and Latency.

Priority Settings for BFD Packets

For all interfaces under over-subscription, the internal priority needs to be assigned to remote BFD Echo packets, so that these BFD packets are not overwhelmed by other data packets. In addition, CoS values need to be set appropriately, so that in the event of an intermediate switch, the reply back of remote BFD Echo packets are protected from all other packets in the switch.

As configured CoS values in ethernet headers may not be retained in Echo messages, CoS values must be explicitly configured in the appropriate egress QoS service policy. CoS values for BFD packets attached to a traffic class can be set using the set cos command. For more information on configuring class-based
unconditional packet marking, see "Configuring Modular QoS Packet Classification" in the *Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Configuration Guide*.

## 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
  - OSPF
  - and OSPFv3
  - MPLS Traffic Engineering (MPLS-TE)
  - Static routes (IPv4 and IPv6)
  - Protocol Independent Multicast (PIM)
  - Hot Standby Router Protocol (HSRP)
  - Virtual Router Redundancy Protocol (VRRP)
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)
  - 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.

Enabling BFD on a Static Route

The following procedure describes how to enable BFD on a static route.

Note: Bundle VLAN sessions are restricted to an interval of 250 milliseconds and a multiplier of 3. More aggressive parameters are not allowed.

SUMMARY STEPS

1. configure
2. router static
3. address-family ipv4 unicast address next-hop bfd fast-detect [minimum-interval interval] [multiplier multiplier]
4. vrf vrf-name
5. address-family ipv4 unicast address next-hop bfd fast-detect
6. commit
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters static route configuration mode, allowing you to configure static routing.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router static</td>
<td>Enables BFD fast-detection on the specified IPV4 unicast destination address prefix and on the forwarding next-hop address.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config)# router static</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family ipv4 unicast address next-hop bfd fast-detect [minimum-interval interval] [multiplier multiplier]</td>
<td>Enables BFD fast-detection on the specified IPV4 unicast destination address prefix and on the forwarding next-hop address.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-static)# address-family ipv4 unicast 0.0.0.0/0 2.6.0.1 bfd fast-detect minimum-interval 1000 multiplier 5</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> vrf vrf-name</td>
<td>Specifies a VPN routing and forwarding (VRF) instance, and enters static route configuration mode for that VRF.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-static)# vrf vrf1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> address-family ipv4 unicast address next-hop bfd fast-detect</td>
<td>Enables BFD fast-detection on the specified IPV4 unicast destination address prefix and on the forwarding next-hop address.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-static-vrf)# address-family ipv4 unicast 0.0.0.0/0 2.6.0.2</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>
BFD for IPv6

Cisco IOS XR software supports bidirectional forwarding detection (BFD) for both IPv4 and IPv6. Bidirectional forwarding detection (BFD) for IPv6 supports the verification of live connectivity on interfaces that use IPv6 addresses.

The live connectivity verification for both IPv4 and IPv6 interfaces is performed by the same services and processes. Both IPv4 and IPv6 BFD sessions can run simultaneously on the same line card.

The same features and configurations that are supported in BFD for IPv4 are also supported in BFD for IPv6.

BFD on Bundled VLANs

BFD for IPv4 on bundled VLANs is supported using static routing, IS-IS, and OSPF. When running a BFD session on a bundled VLAN interface, the BFD session is active as long as the VLAN bundle is up.

As long as the VLAN bundle is active, the following events do not cause the BFD session to fail:

- Failure of a component link.
- Online insertion and removal (OIR) of a line card which hosts one or more of the component links.
- Addition of a component link (by configuration) to the bundle.
- Removal of a component link (by configuration) from the bundle.
- Shutdown of a component link.
- RP switchover.

Note: For more information on configuring a VLAN bundle, see the Configuring Link Bundling on the Cisco ASR 9000 Series Router module.

Keep the following in mind when configuring BFD over bundled VLANs:

- In the case of an RP switchover, configured next-hops are registered in the Routing Information Base (RIB).
- In the case of a BFD restart, static routes remain in the RIB. BFD sessions are reestablished when BFD restarts.

Note: Static BFD sessions are supported on peers with address prefixes whose next-hops are directly connected to the router.
BFD Over Member Links on Link Bundles

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 11: 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** (\texttt{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** (\texttt{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 \texttt{bfd timers nbr-unconfig} is configured on the link, the timer is started. If the BFD configuration is removed on the local router before the timer expires, then the timer is stopped and the behavior is as expected for BFD configuration removal on the local router. If the timer expires, then the behavior is the same as for a BFD session DOWN notification.

- The session startup timer expires before notification from the BFD peer is received.

The BFD session on a bundle member sends BFD state change notifications to the bundle manager. Once BFD state change notifications for bundle member interfaces are received by the bundle manager, the bundle manager determines whether or not the corresponding bundle interface is usable.

A threshold for the minimum number of active member links on a bundle is used by the bundle manager to determine whether the bundle remains active, or is down based on the state of its member links. When BFD is started on a bundle that is already active, the BFD state of the bundle is declared when the BFD state of all the existing active members is known.

Whenever a member’s state changes, the bundle manager determines if the number of active members is less than the minimum number of active links threshold. If so, then the bundle is placed, or remains, in DOWN state. Once the number of active links reaches the minimum threshold then the bundle returns to UP state.
- Another threshold is configurable on the bundle and is used by the bundle manager to determine the minimum amount of active bandwidth to be available before the bundle goes to DOWN state. This is configured using the `bundle minimum-active bandwidth` command.

- The BFD server responds to information from the bundle manager about state changes for the bundle interface and notifies applications on that interface while also sending system messages and MIB traps.

### BFD Multipath Sessions

BFD can be applied over virtual interfaces such as GRE tunnel interfaces, PWHE interfaces, or between interfaces that are multihops away as described in the BFD for MultiHop Paths section. These types of BFD sessions are referred to BFD Multipath sessions.

As long as one path to the destination is active, these events may or may not cause the BFD Multipath session to fail as it depends on the interval negotiated versus the convergence time taken to update forwarding plane:

- Failure of a path
- Online insertion or removal (OIR) of a line card which hosts one or more paths
- Removal of a link (by configuration) which constitutes a path
- Shutdown of a link which constitutes a path

You must configure `bfd multipath include location location-id` command to enable at least one line card for the underlying mechanism that can be used to send and receive packets for the multipath sessions.

If a BFD Multipath session is hosted on a line card that is being removed from the bfd multipath include configuration, online removed, or brought to maintenance mode, then BFD attempts to migrate all BFD Multipath sessions hosted on that line card to another one. In that case, static routes are removed from RIB and then the BFD session is established again and included to RIB.

For more information on PW headend and its configuration, see Implementing Virtual Private LAN Services module in the Cisco ASR 9000 Series Aggregation Services Router L2VPN and Ethernet Services Configuration Guide. For more information on GRE, see Implementing MPLS Layer 2 VPNs module in Cisco ASR 9000 Series Aggregation Services Router L2VPN and Ethernet Services Configuration Guide.

### BFD for MultiHop Paths

BFD multihop (BFD-MH) is a BFD session between two addresses that are not on the same subnet. An example of BFD-MH is a BFD session between PE and CE loopback addresses or BFD sessions between routers that are several TTL hops away. The applications that support BFD multihop are external and internal BGP. BFD multihop supports BFD on arbitrary paths, which can span multiple network hops.

The BFD Multihop feature provides sub-second forwarding failure detection for a destination more than one hop, and up to 255 hops, away. The `bfd multipath 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 over Satellite Interfaces

Bidirectional Forwarding Detection (BFD) over satellite interfaces feature enables BFD support on satellite line cards. Satellite interfaces are known as virtual (bundle) interfaces. BFD uses multipath infrastructure to support BFD on satellite line cards. BFD over satellite is a multipath (MP) single-hop session and is supported on IPv4 address, IPv6 global address, and IPv6 link-local address. The BFD over Satellite is supported only on ASR 9000 Enhanced Ethernet Line Card and is supported in asynchronous mode. BFD over satellite is not supported in echo mode.

Note

- The bfd multipath include location node-id command is not supported on ASR 9000 Ethernet Line Card. Hence, BFD over Satellite Interfaces feature does not work on the ASR 9000 Ethernet Line Card.
- BFD over Satellite Interfaces is not supported on nV Edge system.

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.
  - The maximum rate for BFD sessions per linecard is 9600 pps.
- The maximum number of BFD sessions supported on any one card is 1440.
- The maximum number of members in a bundle is 64.
- The maximum number of BFD sessions on VLANs in a bundle is 128. When using BFD with OSPF, consider the following guidelines:
  - BFD establishes sessions from a neighbor to a designated router (DR) or backup DR (BDR) only when the neighbor state is full.
BFD does not establish sessions between DR-Other neighbors (for example, when their OSPF states are both 2-way).

**Caution**

If you are using BFD with Unicast Reverse Path Forwarding (uRPF) on a particular interface, then you need to use the `echo disable` command to disable echo mode on that interface; otherwise, echo packets will be rejected. For more information, see the Disabling Echo Mode. To enable or disable IPv4 uRPF checking on an IPv4 interface, use the `[no] ipv4 verify unicast source reachable-via` command in interface configuration mode.

---

### Configuring BFD Under a Dynamic Routing Protocol or Using a Static Route

#### Enabling BFD on a BGP Neighbor

BFD can be enabled per neighbor, or per interface. This task describes how to enable BFD for BGP on a neighbor router. To enable BFD per interface, use the steps in the Enabling BFD for OSPF on an Interface.

**Note**

BFD neighbor router configuration is supported for BGP only.

#### SUMMARY STEPS

1. `configure`
2. `router bgp autonomous-system-number`
3. `bfd minimum-interval milliseconds`
4. `bfd multiplier multiplier`
5. `neighbor ip-address`
6. `remote-as autonomous-system-number`
7. `bfd fast-detect`
8. `commit`

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>router bgp autonomous-system-number</code> &lt;br&gt; <strong>Example:</strong>&lt;br&gt; RP/0/RSP0/CPU0:router(config)# router bgp 120</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>bfd minimum-interval milliseconds</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# bfd minimum-interval 6500</td>
</tr>
<tr>
<td></td>
<td>Sets the BFD minimum interval. Range is 15-30000 milliseconds.</td>
</tr>
</tbody>
</table>

| **Step 4** | **bfd multiplier multiplier** |
| Example: | RP/0/RSP0/CPU0:router(config-bgp)# bfd multiplier 7 |
| | Sets the BFD multiplier. |

| **Step 5** | **neighbor ip-address** |
| Example: | RP/0/RSP0/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/RSP0/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/RSP0/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 between the local networking devices and the neighbor 172.168.40.24. |

| **Step 8** | **commit** |

---

**Enabling BFD for OSPF on an Interface**

The following procedures describe how to configure BFD for Open Shortest Path First (OSPF) on an interface. The steps in the procedure are common to the steps for configuring BFD on IS-IS and MPLS-TE; only the command mode differs.

**Note**

BFD per interface configuration is supported for OSPF, OSPFv3, IS-IS, and MPLS-TE only. For information about configuring BFD on an OSPFv3 interface, see [Enabling BFD for OSPFv3 on an Interface](#).
### SUMMARY STEPS

1. configure
2. bfd multipath include 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. commit
10. show run router ospf

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td>(Optional) Enables BFD multipath for the specified bundle on the interface. This step is required for bundle interfaces.</td>
</tr>
</tbody>
</table>
| Step 2 | bfd multipath include location node-id | Enables BFD multipath for the specified bundle on the interface. This step is required for bundle interfaces.  
**Example:** RP/0/RSP0/CPU0:router(config)# bfd multipath include location 0/0/CPU0  
**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 | Enters OSPF configuration mode, allowing you to configure the OSPF routing process. Use the `show ospf` command in EXEC configuration mode to obtain the process-name for the current router.  
**Example:** RP/0/RSP0/CPU0:router(config)# router ospf 0  
**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 | Sets the BFD minimum interval. Range is 15-30000 milliseconds.  
**Example:** RP/0/RSP0/CPU0:router(config-ospf)# bfd minimum-interval 6500  
This example sets the BFD minimum interval to 6500 milliseconds. |
| Step 5 | bfd multiplier multiplier | Sets the BFD multiplier.  
**Example:** RP/0/RSP0/CPU0:router(config-ospf)# bfd multiplier 7  
This example sets the BFD multiplier to 7. |
| Step 6 | area area-id | Configures an Open Shortest Path First (OSPF) area. |
### 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](#).

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>Replace <code>area-id</code> with the OSPF area identifier.</td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-ospf)# area 0</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>Enters interface configuration mode and specifies the interface name and notation <code>rack/slot/module/port</code>.</td>
</tr>
<tr>
<td><strong>interface type interface-path-id</strong></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>• The example indicates a Gigabit Ethernet interface in modular services card slot 3.</td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-ospf-ar)# interface gigabitEthernet 0/3/0/1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>Enables BFD to detect failures in the path between adjacent forwarding engines.</td>
</tr>
<tr>
<td><code>bfd fast-detect</code></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-ospf-ar-if)# bfd fast-detect</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td></td>
</tr>
<tr>
<td><code>commit</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>Verify that BFD is enabled on the appropriate interface.</td>
</tr>
<tr>
<td><code>show run router ospf</code></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-ospf-ar-if)# show run router ospf</code></td>
<td></td>
</tr>
</tbody>
</table>
**SUMMARY STEPS**

1. configure
2. router ospfv3 `process-name`
3. bfd minimum-interval `milliseconds`
4. bfd multiplier `multiplier`
5. area `area-id`
6. interface type `interface-path-id`
7. bfd fast-detect
8. commit
9. show run router ospfv3

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters OSPFv3 configuration mode, allowing you to configure the OSPFv3 routing process.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router ospfv3 <code>process-name</code></td>
<td>Use the <code>show ospfv3</code> command in 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.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-ospfv3)# router ospfv3 0</code></td>
</tr>
<tr>
<td><strong>Step 3</strong> bfd minimum-interval <code>milliseconds</code></td>
<td>Sets the BFD minimum interval. Range is 15-30000 milliseconds. This example sets the BFD minimum interval to 6500 milliseconds.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-ospfv3)# bfd minimum-interval 6500</code></td>
</tr>
<tr>
<td><strong>Step 4</strong> bfd multiplier <code>multiplier</code></td>
<td>Sets the BFD multiplier. This example sets the BFD multiplier to 7.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-ospfv3)# bfd multiplier 7</code></td>
</tr>
<tr>
<td><strong>Step 5</strong> area <code>area-id</code></td>
<td>Configures an OSPFv3 area. Replace <code>area-id</code> with the OSPFv3 area identifier.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-ospfv3)# area 0</code></td>
</tr>
</tbody>
</table>
### Enabling BFD on a Static Route

The following procedure describes how to enable BFD on a static route.

**Note**

Bundle VLAN sessions are restricted to an interval of 250 milliseconds and a multiplier of 3. More aggressive parameters are not allowed.

**SUMMARY STEPS**

1. `configure`
2. `router static`
3. `address-family ipv4 unicast address next-hop bfd fast-detect [minimum-interval interval] [multiplier multiplier]`
4. `vrf vrf-name`
5. `address-family ipv4 unicast address next-hop bfd fast-detect`
6. `commit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
</tr>
</tbody>
</table>
### Command or Action

**Step 2**
- `router static`
  
  **Example:**
  
  RP/0/RSP0/CPU0:router(config)# router static

  Enters static route configuration mode, allowing you to configure static routing.

**Step 3**
- `address-family ipv4 unicast address nexthop bfd fast-detect [minimum-interval interval] [multiplier multiplier]`
  
  **Example:**
  
  RP/0/RSP0/CPU0:router(config-static)# address-family ipv4 unicast 0.0.0.0/0 2.6.0.1 bfd fast-detect minimum-interval 1000 multiplier 5

  Enables BFD fast-detection on the specified IPV4 unicast destination address prefix and on the forwarding next-hop address.
  - Include the optional `minimum-interval` keyword and argument to ensure that the next-hop is assigned with the same hello interval. Replace the `interval` argument with a number that specifies the interval in milliseconds. Range is from 10 through 10000.
  - Include the optional `multiplier` keyword argument to ensure that the next hop is assigned with the same detect multiplier. Replace the `multiplier` argument with a number that specifies the detect multiplier. Range is from 1 through 10.

  **Note** Bundle VLAN sessions are restricted to an interval of 250 milliseconds and a multiplier of 3. More aggressive parameters are not allowed.

**Step 4**
- `vrf vrf-name`
  
  **Example:**
  
  RP/0/RSP0/CPU0:router(config-static)# vrf vrf1

  Specifies a VPN routing and forwarding (VRF) instance, and enters static route configuration mode for that VRF.

**Step 5**
- `address-family ipv4 unicast address nexthop bfd fast-detect`
  
  **Example:**
  
  RP/0/RSP0/CPU0:router(config-static-vrf)# address-family ipv4 unicast 0.0.0.0/0 2.6.0.2

  Enables BFD fast-detection on the specified IPV4 unicast destination address prefix and on the forwarding next-hop address.

**Step 6**
- `commit`

---

### Configuring BFD on Bundle Member Links

#### Prerequisites for Configuring BFD on Bundle Member Links

The physical interfaces that are members of a bundle must be directly connected between peer routers without any switches in between.
Specifying the BFD Destination Address on a Bundle

To specify the BFD destination address on a bundle, complete these steps:

DETAILED STEPS

SUMMARY STEPS

1. configure
2. interface Bundle-Ether | Bundle-POS bundle-id
3. bfd address-family ipv4 destination ip-address
4. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
</tr>
<tr>
<td>Step 2</td>
<td>interface Bundle-Ether</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether 1</td>
</tr>
<tr>
<td>Step 3</td>
<td>bfd address-family ipv4 destination ip-address</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-if)# bfd address-family ipv4 destination 10.20.20.1</td>
</tr>
<tr>
<td>Step 4</td>
<td>commit</td>
</tr>
</tbody>
</table>

Enabling BFD Sessions on Bundle Members

To enable BFD sessions on bundle member links, complete these steps:

SUMMARY STEPS

1. configure
2. interface Bundle-Ether | Bundle-POS bundle-id
3. bfd address-family ipv4 fast-detect
4. commit
DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters interface configuration mode for the specified bundle ID.</td>
</tr>
<tr>
<td><strong>Step 2</strong> interface Bundle-Ether</td>
<td>Enables IPv4 BFD sessions on bundle member links.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> bfd address-family ipv4 fast-detect</td>
<td>Enables IPv4 BFD sessions on bundle member links.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-if)# bfd address-family ipv4 fast-detect</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>

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. commit
### Configuring BFD on Bundle Member Links

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables interface configuration mode for the specified bundle ID.</td>
</tr>
<tr>
<td><strong>Step 2</strong> interface Bundle-Ether <strong>bundle-id</strong></td>
<td>Enters interface configuration mode for the Bundle-Ether interface.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether 1</td>
</tr>
<tr>
<td><strong>Step 3</strong> bundle minimum-active bandwidth <strong>kbps</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000</td>
</tr>
<tr>
<td><strong>Step 4</strong> bundle minimum-active links <strong>links</strong></td>
<td>Sets the number of active links required before a bundle can be brought up or remain up. The range is from 1 to 32.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-if)# bundle minimum-active links 2</td>
</tr>
<tr>
<td><strong>Step 5</strong> commit</td>
<td>Commits the configuration.</td>
</tr>
</tbody>
</table>

### Configuring BFD Packet Transmission Intervals and Failure Detection Times on a Bundle

BFD asynchronous packet intervals and failure detection times for BFD sessions on bundle member links are configured using a combination of the `bfd address-family ipv4 minimum-interval` and `bfd address-family ipv4 multiplier` interface configuration commands on a bundle.

The BFD control packet interval is configured directly using the `bfd address-family ipv4 minimum-interval` command. The BFD echo packet interval and all failure detection times are determined by a combination of the interval and multiplier values in these commands. For more information see the BFD Packet Intervals and Failure Detection.

To configure the minimum transmission interval and failure detection times for BFD asynchronous mode control and echo packets on bundle member links, complete these steps:

DETAILED STEPS
### SUMMARY STEPS

1. configure
2. interface Bundle-Ether [Bundle-POS] bundle-id
3. bfd address-family ipv4 minimum-interval milliseconds
4. bfd address-family ipv4 multiplier multiplier
5. commit

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters interface configuration mode for the specified bundle ID.</td>
</tr>
<tr>
<td><strong>Step 2</strong> interface Bundle-Ether [Bundle-POS] bundle-id</td>
<td>Example: RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether 1</td>
</tr>
<tr>
<td><strong>Step 3</strong> bfd address-family ipv4 minimum-interval milliseconds</td>
<td>Example: RP/0/RSP0/CPU0:router(config-if)# bfd address-family ipv4 minimum-interval 2000</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>• 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 ASR 9000 Series Router is 50 ms.</td>
</tr>
<tr>
<td><strong>Step 4</strong> bfd address-family ipv4 multiplier multiplier</td>
<td>Example: RP/0/RSP0/CPU0:router(config-if)# bfd address-family ipv4 multiplier 30</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>• Although the command allows you to configure a minimum of 2, the supported minimum is 3.</td>
</tr>
<tr>
<td><strong>Step 5</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>
Configuring Allowable Delays for BFD State Change Notifications Using Timers on a Bundle

The BFD system supports two configurable timers to allow for delays in receipt of BFD SCNs from peers before declaring a BFD session on a link bundle member down:

- BFD session startup
- BFD configuration removal by a neighbor

For more information about how these timers work and other BFD state change behavior, see the Overview of BFD State Change Behavior on Member Links and Bundle Status.

To configure the timers that allow for delays in receipt of BFD SCNs from peers, complete these steps:

### SUMMARY STEPS

1. configure
2. interface Bundle-Ether | Bundle-POS] bundle-id
3. bfd address-family ipv4 timers start seconds
4. bfd address-family ipv4 timers nbr-unconfig seconds
5. commit

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters interface configuration mode for the specified bundle ID.</td>
</tr>
<tr>
<td><strong>Step 2</strong> interface Bundle-Ether</td>
<td>Bundle-POS] bundle-id</td>
</tr>
<tr>
<td><strong>Step 3</strong> bfd address-family ipv4 timers start seconds</td>
<td>Specifies the number of seconds after startup of a BFD member link session to wait for the expected notification from the BFD peer to be received, so that the session can be declared up. If the SCN is not received after that period of time, the BFD session is declared down. The range is 60 to 3600. (In Cisco IOS XR Releases 4.0 and 4.0.1, the available minimum is 30, but is not recommended.) <strong>Example:</strong> RP/0/RSP0/CPU0:router(config-if)#</td>
</tr>
<tr>
<td><strong>Step 4</strong> bfd address-family ipv4 timers nbr-unconfig seconds</td>
<td>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. <strong>Example:</strong> RP/0/RSP0/CPU0:router(config-if)#</td>
</tr>
<tr>
<td><strong>Step 5</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>
Enabling Echo Mode to Test the Forwarding Path to a BFD Peer

BFD echo mode is enabled by default for the following interfaces:

- For IPv4 on member links of BFD bundle interfaces.
- For IPv4 on other physical interfaces whose minimum interval is less than two seconds.

Note
If you have configured a BFD minimum interval greater than two seconds on a physical interface using the `bfd minimum-interval` command, then you will need to change the interval to be less than two seconds to support and enable echo mode. This does not apply to bundle member links, which always support echo mode.

Overriding the Default Echo Packet Source Address

If you do not specify an echo packet source address, then BFD uses the IP address of the output interface as the default source address for an echo packet.

In Cisco IOS XR releases before 3.9.0, we recommend that you configure the local router ID using the `router-id` command to change the default IP address for the echo packet source address to the address specified as the router ID.

Beginning in Cisco IOS XR release 3.9.0 and later, you can use the `echo ipv4 source` command in BFD or interface BFD configuration mode to specify the IP address that you want to use as the echo packet source address.

You can override the default IP source address for echo packets for BFD on the entire router, or for a particular interface.

Specifying the Echo Packet Source Address Globally for BFD

To specify the echo packet source IP address globally for BFD on the router, complete the following steps:

**SUMMARY STEPS**

1. `configure`
2. `bfd`
3. `echo ipv4 source ip-address`
4. `commit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
</tr>
</tbody>
</table>
### Specifying the Echo Packet Source Address on an Individual Interface or Bundle

To specify the echo packet source IP address on an individual BFD interface or bundle, complete the following steps:

**SUMMARY STEPS**

1. configure
2. bfd
3. interface type interface-path-id
4. echo ipv4 source ip-address
5. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters BFD configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> bfd</td>
<td>Enters BFD configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# bfd</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> interface type interface-path-id</td>
<td>Enters BFD interface configuration mode for a specific interface or bundle. In BFD interface configuration mode, you can specify an IPv4 address on an individual interface or bundle.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-bfd)# interface gigabitEthernet 0/1/5/0</td>
<td></td>
</tr>
</tbody>
</table>
Configuring BFD Session Teardown Based on Echo Latency Detection

Beginning in Cisco IOS XR 4.0.1, you can configure BFD sessions on non-bundle interfaces to bring down a BFD session when it exceeds the configured echo latency tolerance.

To configure BFD session teardown using echo latency detection, complete the following steps.

Before you enable echo latency detection, be sure that your BFD configuration supports echo mode.

Echo latency detection is not supported on bundle interfaces.

DETAILED STEPS

SUMMARY STEPS

1. configure
2. bfd
3. echo latency detect [percentage percent-value [count packet-count]]
4. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>bfd</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CP00:router(config)# bfd</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>echo latency detect [percentage percent-value [count packet-count]]</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CP00:router(config-bfd)# echo latency detect</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 4</strong></td>
<td>echo ipv4 source ip-address</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CP00:router(config-bfd)# echo ipv4 source 10.10.10.1</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>commit</td>
</tr>
</tbody>
</table>

 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).
Delaying BFD Session Startup Until Verification of Echo Path and Latency

Beginning in Cisco IOS XR Release 4.0.1, you can verify that the echo packet path is working and within configured latency thresholds before starting a BFD session on non-bundle interfaces.

**Note**

Echo startup validation is not supported on bundle interfaces.

To configure BFD echo startup validation, complete the following steps.

**Before You Begin**

Before you enable echo startup validation, be sure that your BFD configuration supports echo mode.

**SUMMARY STEPS**

1. configure
2. bfd
3. echo startup validate [force]
4. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters BFD configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> bfd</td>
<td>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. When the <strong>force</strong> keyword is not configured, the local system performs echo startup validation if the following conditions are true:</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/0RFP0SP0/CPU0:router(config)# bfd</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> echo startup validate [force]</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/0RFP0SP0/CPU0:router(config-bfd)# echo startup validate</td>
<td></td>
</tr>
</tbody>
</table>
### 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**
Disabling Echo Mode on an Individual Interface or Bundle

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

SUMMARY STEPS

1. configure
2. bfd
3. interface type interface-path-id
4. echo disable
5. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>bfd</td>
<td>Enters BFD configuration mode.</td>
</tr>
<tr>
<td>echo disable</td>
<td>Disables echo mode on the router.</td>
</tr>
<tr>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

Example:
```
RP/0/RSP0/CPU0:router(config)# bfd
RP/0/RSP0/CPU0:router(config-bfd)# echo disable
```
### Minimizing BFD Session Flapping Using BFD Dampening

To configure BFD dampening to control BFD session flapping, complete the following steps.

**SUMMARY STEPS**

1. configure
2. bfd
3. dampening [bundle-member] {initial-wait | maximum-wait | secondary-wait} milliseconds
4. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>bfd</td>
<td>Enters BFD configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# bfd</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>dampening [bundle-member] {initial-wait</td>
<td>maximum-wait</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-bfd)# dampening</td>
<td></td>
</tr>
</tbody>
</table>
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 on the entire router, or for a particular interface. These sections describe about:

Note

The command-line interface (CLI) is slightly different in BFD configuration and BFD interface configuration. For BFD configuration, the `disable` keyword is not optional. Therefore, to enable BFD configuration in that mode, you need to use the `no` form of the command.

---

Enabling and Disabling IPv6 Checksum Calculations for BFD on a Router

To enable or disable IPv6 checksum calculations globally on the router complete the following steps:

**SUMMARY STEPS**

1. configure
2. bfd
3. ipv6 checksum [disable]
4. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 bfd</td>
<td>Enters BFD configuration mode.</td>
</tr>
</tbody>
</table>

Example:

```
RP/0/RSP0/CPU0:router(config)# bfd
```
### 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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters BFD configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> bfd</td>
<td>Enters BFD interface configuration mode for a specific interface or bundle.</td>
</tr>
<tr>
<td><strong>Step 3</strong> interface type interface-path-id</td>
<td>Enters BFD interface configuration mode for a specific interface or bundle.</td>
</tr>
</tbody>
</table>

**SUMMARY STEPS**

1. configure
2. bfd
3. interface type interface-path-id
4. ipv6 checksum [disable]
5. commit
### Clearing and Displaying BFD Counters

The following procedure describes how to display and clear BFD packet counters. You can clear packet counters for BFD sessions that are hosted on a specific node or on a specific interface.

**SUMMARY STEPS**

1. `show bfd counters [ipv4 | 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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Displays the BFD counters for IPv4 packets, IPv6 packets, or all packets.</td>
</tr>
<tr>
<td>`show bfd counters [ipv4</td>
<td>all] packet interface type interface-path-id] location node-id`</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router# show bfd counters all packet location 0/3/cpu0</code></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Clears the BFD counters for IPv4 packets, IPv6 packets, or all packets.</td>
</tr>
<tr>
<td>`clear bfd counters [ipv4</td>
<td>ipv6</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router# clear bfd counters all packet location 0/3/cpu0</code></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Verifies that the BFD counters for IPv4 packets, IPv6 packets, or all packets have been cleared.</td>
</tr>
<tr>
<td>`show bfd counters [ipv4</td>
<td>ipv6</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router# show bfd counters all packet location 0/3/cpu0</code></td>
</tr>
</tbody>
</table>
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/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#router bgp 65000
RP/0/RSP0/CPU0:router(config-bgp)#bfd multiplier 2
RP/0/RSP0/CPU0:router(config-bgp)#bfd minimum-interval 20
RP/0/RSP0/CPU0:router(config-bgp)#neighbor 192.168.70.24
RP/0/RSP0/CPU0:router(config-bgp-nbr)#remote-as 2
RP/0/RSP0/CPU0:router(config-bgp-nbr)#bfd fast-detect
RP/0/RSP0/CPU0:router(config-bgp-nbr)#commit
RP/0/RSP0/CPU0:router(config-bgp-nbr)#end
RP/0/RSP0/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/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#router ospf 0
RP/0/RSP0/CPU0:router(config-ospf)#area 0
RP/0/RSP0/CPU0:router(config-ospf-area)#interface gigabitEthernet 0/3/0/1
RP/0/RSP0/CPU0:router(config-ospf-area-if)#bfd fast-detect
RP/0/RSP0/CPU0:router(config-ospf-area-if)#commit
RP/0/RSP0/CPU0:router(config-ospf-area-if)#end

RP/0/RSP0/CPU0:router#show run router ospf
router ospf 0
area 0
interface GigabitEthernet0/3/0/1
    bfd fast-detect
```

The following example shows how to enable BFD for OSPFv3 on a Gigabit Ethernet interface:

```
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#router ospfv3 0
RP/0/RSP0/CPU0:router(config-ospfv3)#bfd minimum-interval 6500
RP/0/RSP0/CPU0:router(config-ospfv3)#bfd multiplier 7
RP/0/RSP0/CPU0:router(config-ospfv3-area)#area 0
RP/0/RSP0/CPU0:router(config-ospfv3-area)#interface gigabitethernet 0/1/5/0
RP/0/RSP0/CPU0:router(config-ospfv3-area-if)#bfd fast-detect
RP/0/RSP0/CPU0:router(config-ospfv3-area-if)#commit
RP/0/RSP0/CPU0:router(config-ospfv3-area-if)#end

RP/0/RSP0/CPU0:router#show run router ospfv3
router ospfv3
area 0
interface GigabitEthernet0/1/5/0
    bfd fast-detect
```
BFD Over Static Routes: Examples

The following example shows how to enable BFD on an IPv4 static route. In this example, BFD sessions are established with the next-hop 10.3.3.3 when it becomes reachable.

```plaintext
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#router static
RP/0/RSP0/CPU0:router(config-static)#address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-static)#10.2.2.0/24 10.3.3.3 bfd fast-detect
RP/0/RSP0/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.

```plaintext
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#router static
RP/0/RSP0/CPU0:router(config-static)#address-family ipv6 unicast
RP/0/RSP0/CPU0:router(config-static)#end
RP/0/RSP0/CPU0:router#show run router static address-family ipv6 unicast
```

BFD on Bundled VLANs: Example

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

```plaintext
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/CPU0:router(config-bfd)#multipath include location 0/0/CPU0
RP/0/RSP0/CPU0:router(config-bfd)#exit

RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#interface Bundle-ether 1
RP/0/RSP0/CPU0:router(config-if)#bundle maximum-active links 1
RP/0/RSP0/CPU0:router(config-if)#exit

RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#interface TenGigE 0/1/0/1
RP/0/RSP0/CPU0:router(config-if)#bundle id 1 mode active
RP/0/RSP0/CPU0:router(config-if)#exit

RP/0/RSP0/CPU0:router(config)#interface TenGigE 0/2/0/1
RP/0/RSP0/CPU0:router(config-if)#bundle id 1 mode active
RP/0/RSP0/CPU0:router(config-if)#exit

RP/0/RSP0/CPU0:router(config)#router static
RP/0/RSP0/CPU0:router(config-static)#address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-static-afi)#10.2.1.0/24 172.16.1.2 bfd fast-detect minimum-interval 250
RP/0/RSP0/CPU0:router(config-static-afi)#10.2.2.0/24 172.16.2.2 bfd fast-detect minimum-interval 250
RP/0/RSP0/CPU0:router(config-static-afi)#10.2.3.0/24 172.16.3.2 bfd fast-detect minimum-interval 250
RP/0/RSP0/CPU0:router(config-static-afi)#exit

RP/0/RSP0/CPU0:router(config)#interface Bundle-Ether1.2
RP/0/RSP0/CPU0:router(config-if)#ipv4 address 172.16.2.1 255.255.255.0
RP/0/RSP0/CPU0:router(config-if)#encapsulation dot1q 2
RP/0/RSP0/CPU0:router(config-if)#exit

RP/0/RSP0/CPU0:router(config)#interface Bundle-Ether1.1
```
The following example shows how to configure BFD on member links of Ethernet bundle interfaces:

```
bfd
tinterface Bundle-Ether4
  echo disable
  !
tinterface GigabitEthernet0/0/0/3
  echo disable
  !
tinterface GigabitEthernet0/0/0/4
  bundle id 1 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
  !
tinterface Bundle-Ether1.1
  ipv4 address 192.168.100.1/30
  encapsulation dot1q 1001
  !
tinterface 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
  !
tinterface 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
  !
tinterface 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
  !
tinterface GigabitEthernet 0/0/0/2
  ipv4 address 192.168.10.1/30
  !
tinterface GigabitEthernet 0/0/0/2.1
  ipv4 address 192.168.11.1/30
  ipv6 address beef:cafe::1/64
  encapsulation dot1q 2001
  !
tinterface GigabitEthernet 0/0/0/2.2
  ipv4 address 192.168.12.1/30
  encapsulation dot1q 2002
  !
tinterface GigabitEthernet 0/0/0/2.3
  ipv4 address 192.168.13.1/30
  encapsulation dot1q 2003
  !
router static
```
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:

```bash
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/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:

```bash
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/CPU0:router(config-bfd)#interface gigabitethernet 0/1/0/0
RP/0/RSP0/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:

```bash
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/CPU0:router(config-bfd)#interface pos 0/1/0/0
RP/0/RSP0/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:

```bash
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/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:

```bash
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/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/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/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/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/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/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/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/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/CPU0:router(config-bfd)#echo disable
```

The following example shows how to disable echo mode on an interface:

```
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/CPU0:router(config-bfd)#interface gigabitethernet 0/1/0/0
RP/0/RSP0/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/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/CPU0:router(config-bfd)#dampening bundle-member initial-wait 8000
RP/0/RSP0/CPU0:router(config-bfd)#dampening bundle-member maximum-wait 15000
```
BFD IPv6 Checksum: Examples

The following example shows how to change the default initial-wait for BFD on a non-bundle interface:

```bash
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
dampening initial-wait 30000
RP/0/RSP0/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:

```bash
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/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:

```bash
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/CPU0:router(config-bfd)#no ipv6 checksum disable
```

The following example shows how to enable echo mode for BFD sessions on an individual interface:

```bash
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/CPU0:router(config-bfd)#interface gigabitethernet 0/1/0/0
RP/0/RSP0/CPU0:router(config-bfd-if)#ipv6 checksum
```

The following example shows how to disable echo mode for BFD sessions on an individual interface:

```bash
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#bfd
RP/0/RSP0/CPU0:router(config-bfd)#interface gigabitethernet 0/1/0/0
RP/0/RSP0/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)**

```bash
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#interface GigabitEthernet8/1/0
RP/0/RSP0/CPU0:router(config-if)#description to-TestBed1 G0/0/0/0
RP/0/RSP0/CPU0:router(config-if)#ip address 192.0.2.2 255.255.255.0
RP/0/RSP0/CPU0:router(config-if)#bfd interval 100 min_rx 100 multiplier 3
RP/0/RSP0/CPU0:router(config-if)#bfd neighbor 192.0.2.1
```

**Router 2 (Cisco IOS XR Software)**

```bash
RP/0/RSP0/CPU0:router#configure
```
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:


- 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

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFD commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td>Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference</td>
</tr>
<tr>
<td>Configuring QoS packet classification</td>
<td>Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Configuration Guide</td>
</tr>
</tbody>
</table>

Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>
## RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>rfc5880_bfd_base</td>
<td><em>Bidirectional Forwarding Detection</em>, June 2010</td>
</tr>
<tr>
<td>rfc5881_bfd_ipv4_ipv6</td>
<td><em>BFD for IPv4 and IPv6 (Single Hop)</em>, June 2010</td>
</tr>
<tr>
<td>rfc5883_bfd_multihop</td>
<td><em>BFD for Multihop Paths</em>, June 2010</td>
</tr>
</tbody>
</table>

## MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></td>
</tr>
</tbody>
</table>

## Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a></td>
</tr>
</tbody>
</table>
Implementing EIGRP

The Enhanced Interior Gateway Routing Protocol (EIGRP) is an enhanced version of IGRP developed by Cisco. This module describes the concepts and tasks you need to implement basic EIGRP configuration using Cisco IOS XR software. EIGRP uses distance vector routing technology, which specifies that a router need not know all the router and link relationships for the entire network. Each router advertises destinations with a corresponding distance and upon receiving routes, adjusts the distance and propagates the information to neighboring routes.

For EIGRP configuration information related to the following features, see the Related Documents, on page 228 section of this module.

- Multiprotocol Label Switching (MPLS) Layer 3 Virtual Private Network (VPN)
- Site of Origin (SoO) Support

Note

For more information about EIGRP on the Cisco IOS XR software and complete descriptions of the EIGRP commands listed in this module, see the EIGRP Commands chapter in the Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference. 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

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.7.2</td>
<td>This feature was introduced.</td>
</tr>
<tr>
<td>Release 6.0.1</td>
<td>Support for Site of origin (SoO) attribute was added</td>
</tr>
</tbody>
</table>

- Prerequisites for Implementing EIGRP, page 198
- Restrictions for Implementing EIGRP, page 198
- Information About Implementing EIGRP, page 198
- How to Implement EIGRP, page 210
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.
- Bidirectional Forwarding Detection (BFD) feature and the Simple Network Management Protocol (SNMP) MIB are not supported.
- Interface static routes are not automatically redistributed into EIGRP, because there are no network commands.
- Metric configuration (either through the `default-metric` command or a route policy) is required for redistribution of connected and static routes.
- Auto summary is disabled by default.
- Stub leak maps are not supported.

Information About Implementing EIGRP

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

EIGRP Functional Overview

Enhanced Interior Gateway Routing Protocol (EIGRP) is an interior gateway protocol suited for many different topologies and media. EIGRP scales well and provides extremely quick convergence times with minimal network traffic.

EIGRP has very low usage of network resources during normal operation. Only hello packets are transmitted on a stable network. When a change in topology occurs, only the routing table changes are propagated and not the entire routing table. Propagation reduces the amount of load the routing protocol itself places on the network. EIGRP also provides rapid convergence times for changes in the network topology.

The distance information in EIGRP is represented as a composite of available bandwidth, delay, load utilization, and link reliability with improved convergence properties and operating efficiency. The fine-tuning of link characteristics achieves optimal paths.
The convergence technology that EIGRP uses is based on research conducted at SRI International and employs an algorithm referred to as the Diffusing Update Algorithm (DUAL). This algorithm guarantees loop-free operation at every instant throughout a route computation and allows all devices involved in a topology change to synchronize at the same time. Routers that are not affected by topology changes are not involved in recomputations. The convergence time with DUAL rivals that of any other existing routing protocol.

**EIGRP Features**

EIGRP offers the following features:

- **Fast convergence**—The DUAL algorithm allows routing information to converge as quickly as any currently available routing protocol.

- **Partial updates**—EIGRP sends incremental updates when the state of a destination changes, instead of sending the entire contents of the routing table. This feature minimizes the bandwidth required for EIGRP packets.

- **Neighbor discovery mechanism**—This is a simple hello mechanism used to learn about neighboring routers. It is protocol independent.

- **Variable-length subnet masks (VLSMs).**

- **Arbitrary route summarization.**

- **Scaling**—EIGRP scales to large networks.

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

- Provider Edge (PE)-Customer Edge (CE) protocol support with Site of Origin (SoO) and Border Gateway Protocol (BGP) cost community support.

- PECE protocol support for MPLS.

**EIGRP Components**

EIGRP has the following four basic components:

- Neighbor discovery or neighbor recovery

- Reliable transport protocol

- DUAL finite state machine

- Protocol-dependent modules

Neighbor discovery or neighbor recovery is the process that routers use to dynamically learn of other routers on their directly attached networks. Routers must also discover when their neighbors become unreachable or inoperative. Neighbor discovery or neighbor recovery is achieved with low overhead by periodically sending small hello packets. As long as hello packets are received, the Cisco IOS XR software can determine that a neighbor is alive and functioning. After this status is determined, the neighboring routers can exchange routing information.

The reliable transport protocol is responsible for guaranteed, ordered delivery of EIGRP packets to all neighbors. It supports intermixed transmission of multicast and unicast packets. Some EIGRP packets must be sent reliably and others need not be. For efficiency, reliability is provided only when necessary. For example, on
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:

```
RP/0/RSP0/CPU0:router# configuration
RP/0/RSP0/CPU0:router(config)# router eigrp 100
RP/0/RSP0/CPU0:router(config-eigrp)#
```

**VRF Configuration Mode**

The following example shows how to enter VRF configuration mode:

```
RP/0/RSP0/CPU0:router# configuration
RP/0/RSP0/CPU0:router(config)# router eigrp 100
RP/0/RSP0/CPU0:router(config-eigrp)# vrf customer1
```
IPv4 Address Family Configuration Mode

The following example shows how to enter IPv4 address family configuration mode:

```
RP/0/RSP0/CPU0:router# configuration
RP/0/RSP0/CPU0:router(config)# router eigrp 100
RP/0/RSP0/CPU0:router(config-eigrp)# address-family ipv4
RP/0/RSP0/CPU0:router(config-eigrp-af)#
```

IPv4 VRF Address Family Configuration Mode

The following example shows how to enter IPv4 VRF address family configuration mode:

```
RP/0/RSP0/CPU0:router# configuration
RP/0/RSP0/CPU0:router(config)# router eigrp 100
RP/0/RSP0/CPU0:router(config-eigrp)# vrf customer1
RP/0/RSP0/CPU0:router(config-eigrp-vrf)# address-family ipv4
RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)#
```

Interface Configuration Mode

The following example shows how to enter interface configuration mode in IPv4 address family configuration mode:

```
RP/0/RSP0/CPU0:router# configuration
RP/0/RSP0/CPU0:router(config)# router eigrp 100
RP/0/RSP0/CPU0:router(config-eigrp)# address-family ipv4
RP/0/RSP0/CPU0:router(config-eigrp-af)# interface GigabitEthernet 0/3/0/0
RP/0/RSP0/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. When redistribution is configured under a VRF, EIGRP retrieves extended
communities attached to the route in the routing information base (RIB). The SoO is used to filter out routing loops in the presence of MPSL VPN backdoor links.

### Metric Weights for EIGRP Routing

EIGRP uses the minimum bandwidth on the path to a destination network and the total delay to compute routing metrics. You can use the `metric weights` command to adjust the default behavior of EIGRP routing and metric computations. For example, this adjustment allows you to tune system behavior to allow for satellite transmission. EIGRP metric defaults have been carefully selected to provide optimal performance in most networks.

By default, the EIGRP composite metric is a 32-bit quantity that is a sum of the segment delays and lowest segment bandwidth (scaled and inverted) for a given route. For a network of homogeneous media, this metric reduces to a hop count. For a network of mixed media (FDDI, Ethernet, and serial lines running from 9600 bits per second to T1 rates), the route with the lowest metric reflects the most desirable path to a destination.

### Mismatched K Values

Mismatched K values (EIGRP metrics) can prevent neighbor relationships from being established and can negatively impact network convergence. The following example explains this behavior between two EIGRP peers (ROUTER-A and ROUTER-B).

The following error message is displayed in the console of ROUTER-B because the K values are mismatched:

```
RP/0/RSP0/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 $k_1$ argument to adjust the bandwidth calculation. The value of 1 is entered for the $k_3$ argument to adjust the delay calculation.

```plaintext
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.

```plaintext
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/RSP0/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/RSP0/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
```

The receipt of a goodbye message by a non-supporting peer does not disrupt normal network operation. The non-supporting 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 12: Floating Summary Route Is Applied to Router-B**, on page 204 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/RSP0/CPU0:router(config)# router eigrp 100
RP/0/RSP0/CPU0:router(config-eigrp)# address-family ipv4
RP/0/RSP0/CPU0:router(config-eigrp-af)# interface GigabitEthernet 0/3/0/0
RP/0/RSP0/CPU0:router(config-eigrp-af-if)# summary-address 100.0.0.0 0.0.0.0
```
Figure 12: Floating Summary Route Is Applied to Router-B

```
RP/0/RP0/CPU0:router-B# show route

0.0.0.0/0 via <Router-A> (489765/170)
```

```
RP/0/RP0/CPU0:router-C# show route

0.0.0.0/0 via <Router-B> (489765/90)
```

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 0.0.0.0.0/0 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 0.0.0.0.0/0, 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 13: Floating Summary Route Applied for Dual-Homed Remotes, on page 205 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 13: 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.

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 14: Simple Hub-and-Spoke Network, on page 206, 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:

```bash
RP/0/RSP0/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:

```bash
RP/0/RSP0/CPU0:router(config-rpl)# add eigrp-metric bandwidth delay reliability loading mtu
```

A route policy can be used in EIGRP only if all the statements are applicable to the particular EIGRP attach point. The following commands accept a route policy:

- **default-information allowed**—Match statements are allowed for destination. No set statements are allowed.
- **route-policy**—Match statements are allowed for destination, next hop, and tag. Set statements are allowed for eigrp-metric and tag.
- **redistribute**—Match statements are allowed for destination, next hop, source-protocol, tag and route-type. Set statements are allowed for eigrp-metric and tag.

The range for setting a tag is 0 to 255 for internal routes and 0 to 4294967295 for external routes.

### EIGRP Layer 3 VPN PE-CE Site-of-Origin

The EIGRP MPLS and IP VPN PE-CE Site-of-Origin (SoO) feature introduces the capability to filter Multiprotocol Label Switching (MPLS) and IP Virtual Private Network (VPN) traffic on a per-site basis for EIGRP networks. SoO filtering is configured at the interface level and is used to manage MPLS and IP VPN traffic and to prevent transient routing loops from occurring in complex and mixed network topologies.

#### Router Interoperation with the Site-of-Origin Extended Community

The configuration of the SoO extended community allows routers that support this feature to identify the site from which each route originated. When this feature is enabled, the EIGRP routing process on the PE or CE router checks each received route for the SoO extended community and filters based on the following conditions:

- A received route from BGP or a CE router contains a SoO value that matches the SoO value on the receiving interface:
  - If a route is received with an associated SoO value that matches the SoO value that is configured on the receiving interface, the route is filtered out because it was learned from another PE router or from a backdoor link. This behavior is designed to prevent routing loops.

- A received route from a CE router is configured with a SoO value that does not match:
  - If a route is received with an associated SoO value that does not match the SoO value that is configured on the receiving interface, the route is accepted into the EIGRP topology table so that it can be redistributed into BGP.
  - If the route is already installed in the EIGRP topology table but is associated with a different SoO value, the SoO value from the topology table is used when the route is redistributed into BGP.

- A received route from a CE router does not contain a SoO value:
If a route is received without a SoO value, the route is accepted into the EIGRP topology table, and the SoO value from the interface that is used to reach the next-hop CE router is appended to the route before it is redistributed into BGP.

When BGP and EIGRP peers that support the SoO extended community receive these routes, they also receive the associated SoO values and pass them to other BGP and EIGRP peers that support the SoO extended community. This filtering is designed to prevent transient routes from being relearned from the originating site, which prevents transient routing loops from occurring.

In conjunction with BGP cost community, EIGRP, BGP, and the RIB ensure that paths over the MPLS VPN core are preferred over backdoor links.

For MPLS and IP VPN and SoO configuration information, see Implementing MPLS Layer 3 VPNs in the Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide.

### Route Manipulation using SoO match condition

The SoO configuration in EIGRP network can be used to manipulate routes using the SoO match condition in the routing policy. The egress interface of a PE router is used to compare and manipulate routes based on the SoO configuration on the remote PE router.

**Topology**

In the following topology, CE1, CE2 and CE3 are the customer edge routers. PE1 and PE2 are the provider edge routers. By default, CE1 will use PE1->PE2 to reach CE3. To configure CE1 to use CE2 to reach CE3, the metric advertised by PE1 must be increased.

The routing policy on PE1 manipulates routes received from CE3 via PE2, by using the SoO match condition. With this feature added, PE1 can increase the metric while advertising routes to CE1.

```
/*SoO tag is assigned on PE2 router*/

router(config)#interface GigabitEthernet0/0/0/11
router(config-if)#site-of-origin 33.33.33.33:33
```

---

Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x
/* A route-policy defined on PE1 */

router(config)#route-policy test
router(config-rpl)#if extcommunity soo matches-any (33.33.33.33:33) then
router(config-rpl-if)#set eigrp-metric 2121212121 333333333 245 250 1455
router(config-rpl-if)#endif
router(config-rpl-if)#end-policy
router (config)#commit
router (config)#

router(config)#interface GigabitEthernet0/3/0/1
router (config-if)#route-policy test out

Verification:

/*A route with poor metric advertised by PE1 is installed into CE1's routing table for SoO of site C3. */

router#show eigrp topology 6:6::1/128

IPv6-EIGRP AS(100): Topology entry for 6:6::1/128
 State is Passive, Query origin flag is 1, 1 Successor(s), FD is 15539149614794, RIB is 4294967295 Routing Descriptor Blocks: fe80::226:98ff:fe24:5109 (GigabitEthernet0/0/0/15), from fe80::226:98ff:fe24:5109, Send flag is 0x0
 Composite metric is (15539149614794/15539148304382), Route is Internal Vector metric:
 Minimum bandwidth is 1000000 Kbit
 Total delay is 237108596182784 picoseconds
 Load is 250/255
 Minimum MTU is 1455
 Hop count is 2
 Originating router is 2.2.2.2
Extended Community:
 SoO:33.33.33.33:33

Note:
This feature is applicable to both ipv4 as well as ipv6.
All types of SoO(IP-Address, ASN2, ASN4) are supported.

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:
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. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Specifies the autonomous system number of the routing process to configure an EIGRP routing process.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router eigrp <em>as-number</em></td>
<td>Specifies the autonomous system number of the routing process to configure an EIGRP routing process.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router eigrp 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family { ipv4 }</td>
<td>Enters an address family configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-eigrp)# address-family ipv4</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> router-id <em>id</em></td>
<td>(Optional) Configures a router-id for an EIGRP process.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>It is good practice to use the <code>router-id</code> 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.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>default-metric bandwidth delay reliability loading mtu</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>distance internal-distance external-distance</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>interface type interface-path-id</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>holdtime seconds</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>To ensure nonstop forwarding during RP failovers, as the number of neighbors increase, a higher holdtime than the default value is recommended. With 256 neighbors across all VRFs, we recommend 60 seconds.</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>bandwidth-percent percent</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>commit</td>
</tr>
</tbody>
</table>
Before You Begin

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. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Specifies the AS number of the routing process to configure an EIGRP routing process</td>
</tr>
<tr>
<td><strong>Step 2</strong> router eigrp  as-number</td>
<td>Enters an address family configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router eigrp 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family { ipv4 }</td>
<td>Applies a routing policy to updates advertised to or received from an EIGRP neighbor.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-eigrp)# address-family ipv4</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> route-policy name out</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-eigrp-af)# route-policy FILTER_DEFAULT out</td>
<td></td>
</tr>
</tbody>
</table>
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 }
4. redistribute { { bgp | connected | isis | ospf | rip | static } [ as-number ] | route-policy name }
5. redistribute maximum-prefix maximum [ threshold ] [ [ dampened ] [ reset-time minutes ] [ restart minutes ] [ restart-count number ] | [ warning-only ]]
6. timers nsf route-hold seconds
7. maximum paths maximum
8. maximum-prefix maximum [ threshold ] [ [ dampened ] [ reset-time minutes ] [ restart minutes ] [ restart-count number ] | [ warning-only ]]
9. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>2</td>
<td>router eigrp <em>as-number</em></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# router eigrp 100</td>
</tr>
<tr>
<td>3</td>
<td>address-family { ipv4 }</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-eigrp)# address-family ipv4</td>
</tr>
<tr>
<td>4</td>
<td>redistribute {{ bgp</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-eigrp-af)# redistribute bgp 100</td>
</tr>
<tr>
<td>5</td>
<td>redistribute maximum-prefix <em>maximum</em> [ threshold ] [[ dampened ] [ reset-time <em>minutes</em> ] [ restart <em>minutes</em> ] [ restart-count <em>number</em> ] [ warning-only ]]</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-eigrp-af)# redistribute maximum-prefix 5000 95 warning-only</td>
</tr>
<tr>
<td>6</td>
<td>timers nsf route-hold <em>seconds</em></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-eigrp-af)# timers nsf route-hold 120</td>
</tr>
<tr>
<td>7</td>
<td>maximum paths <em>maximum</em></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-eigrp-af)# maximum paths 10</td>
</tr>
<tr>
<td>8</td>
<td>maximum-prefix <em>maximum</em> [ threshold ] [[ dampened ] [ reset-time <em>minutes</em> ] [ restart <em>minutes</em> ] [ restart-count <em>number</em> ] [ warning-only ]]</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-eigrp-af)# maximum-prefix 50000</td>
</tr>
<tr>
<td>9</td>
<td>commit</td>
</tr>
</tbody>
</table>
Creating a Route Policy and Attaching It to an EIGRP Process

This task defines a route policy and shows how to attach it to an EIGRP process.

A route policy definition consists of the route-policy command and name argument followed by a sequence of optional policy statements, and then closed with the end-policy command.

A route policy is not useful until it is applied to routes of a routing protocol.

SUMMARY STEPS

1. configure
2. route-policy name
3. set eigrp-metric bandwidth delay reliability load mtu
4. end-policy
5. commit
6. configure
7. router eigrp as-number
8. address-family { ipv4 }
9. route-policy route-policy-name { in | out }
10. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Defines a route policy and enters route-policy configuration mode.</td>
</tr>
<tr>
<td>Step 2 route-policy name</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# route-policy IN-IPv4</td>
</tr>
<tr>
<td>Step 3 set eigrp-metric bandwidth delay reliability load mtu</td>
<td>(Optional) Sets the EIGRP metric attribute.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-rpl)# set eigrp metric 42 100 200 100 1200</td>
</tr>
<tr>
<td>Step 4 end-policy</td>
<td>Ends the definition of a route policy and exits route-policy configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-rpl)# end-policy</td>
</tr>
<tr>
<td>Step 5 commit</td>
<td></td>
</tr>
<tr>
<td>Step 6 configure</td>
<td></td>
</tr>
</tbody>
</table>
### 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>router eigrp as-number</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config)# router eigrp 100</code></td>
</tr>
<tr>
<td></td>
<td>Specifies the autonomous system number of the routing process to configure an EIGRP routing process.</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>address-family { ipv4 }</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-eigrp)# address-family ipv4</code></td>
</tr>
<tr>
<td></td>
<td>Enters an address family configuration mode.</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>**route-policy route-policy-name { in</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-eigrp-af)# route-policy IN-IPv4 in</code></td>
</tr>
<tr>
<td></td>
<td>Applies a routing policy to updates advertised to or received from an EIGRP neighbor.</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td><strong>commit</strong></td>
</tr>
</tbody>
</table>

#### SUMMARY STEPS

1. configure
2. `router eigrp as-number`
3. `address-family { ipv4 }`
4. `stub [ receive-only | [ [ connected ] [ redistributed ] [ static ] [ summary ] ] ]`
5. commit
6. `show eigrp [ ipv4 ] neighbors [ as-number ] [ detail ] [ type interface-path-id | static ]`

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.
DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Specifies the autonomous system number of the routing process to configure an EIGRP routing process.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router eigrp <code>as-number</code></td>
<td>Enters an address family configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config)# router eigrp 100</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family <code>{ ipv4 }</code></td>
<td>Configures a router as a stub for EIGRP.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-eigrp)# address-family ipv4</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> <code>stub</code> <code>[ receive-only ] [ [ connected ] [ redistributed ] [ static ] [ summary ] ]</code></td>
<td>Verifies that a remote router has been configured as a stub router with EIGRP.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-eigrp-af)# stub receive-only</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> commit</td>
<td>The last line of the output shows the stub status of the remote or spoke router.</td>
</tr>
<tr>
<td><strong>Step 6</strong> show eigrp <code>[ ipv4 ]</code> neighbors `[ as-number ] [ detail ] [ type interface-path-id</td>
<td>static ]`</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router# show eigrp neighbors detail</code></td>
<td></td>
</tr>
</tbody>
</table>

Configuring EIGRP as a PE-CE Protocol

Perform this task to configure EIGRP on the provider edge (PE) and establish provider edge-to-customer edge (PE-CE) communication using EIGRP.
SUMMARY STEPS

1. configure
2. router eigrp as-number
3. vrf vrf-name
4. address-family { ipv4 }
5. router-id router-id
6. autonomous-system as-number
7. redistribute {{ bgp | connected | isis | ospf | ospfv3 | rip | static } [ as-number | instance-name ]} { route-policy name }
8. interface type interface-path-id
9. site-of-origin { as-number:number | ip-address :number }
10. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router eigrp as-number</td>
<td>Specifies the autonomous system number of the routing process to configure an EIGRP routing process</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router eigrp 100</td>
<td></td>
</tr>
<tr>
<td>Step 3 vrf vrf-name</td>
<td>Configures a VPN routing and forwarding (VRF) instance.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-eigrp)# vrf vrf_A</td>
<td></td>
</tr>
<tr>
<td>Step 4 address-family { ipv4 }</td>
<td>Enters a VRF address family configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-eigrp-vrf)# address-family ipv4</td>
<td></td>
</tr>
<tr>
<td>Step 5 router-id router-id</td>
<td>Configures a router ID for the EIGRP process.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# router-id 33</td>
<td></td>
</tr>
<tr>
<td>Step 6 autonomous-system as-number</td>
<td>Configures an EIGRP routing process to run within the VRF instance.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# autonomous-system 2</td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td>You must configure the autonomous system under VRF configuration to bring-up the VRF interface.</td>
</tr>
</tbody>
</table>
### Redistributing BGP Routes into EIGRP

Perform this task to redistribute BGP routes into EIGRP.

Typically, EIGRP routes are redistributed into BGP with extended community information appended to the route. BGP carries the route over the VPN backbone with the EIGRP-specific information encoded in the BGP extended community attributes. After the peering customer site receives the route, EIGRP redistributes the BGP route then extracts the BGP extended community information and reconstructs the route as it appeared in the original customer site.

When redistributing BGP routes into EIGRP, the receiving provider edge (PE) EIGRP router looks for BGP extended community information. If the information is received, it is used to recreate the original EIGRP route. If the information is missing, EIGRP uses the configured default metric value.

If the metric values are not derived from the BGP extended community and a default metric is not configured, the route is not advertised to the customer edge (CE) router by the PE EIGRP. When BGP is redistributed into BGP, metrics may not be added to the BGP prefix as extended communities; for example, if EIGRP is not running on the other router. In this case, EIGRP is redistributed into BGP with a “no-metrics” option.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>redistribute</strong> `{{ bgp</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# redistribute bgp 100</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>interface</strong> <code>type interface-path-id</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# interface gigabitEthernet 0/1/5/0</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td><strong>site-of-origin</strong> ` { as-number:number</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-eigrp-vrf-af-if)# site-of-origin 3:4</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td><strong>commit</strong></td>
</tr>
</tbody>
</table>

**Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x**
SUMMARY STEPS

1. configure
2. router eigrp as-number
3. vrf vrf-name
4. address-family { ipv4 }
5. redistribute { { bgp | connected | isis | ospf | ospfv3 | rip | static } [ as-number | instance-name ] [ route-policy name ] }
6. route-policy route-policy-name { in | out }
7. default-metric bandwidth delay reliability loading mtu
8. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router eigrp as-number</td>
<td>Specifies the autonomous system number of the routing process to configure an EIGRP routing process.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# router eigrp 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> vrf vrf-name</td>
<td>Configures a VRF instance.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-eigrp)# router eigrp 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> address-family { ipv4 }</td>
<td>Enters a VRF address family configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-eigrp-vrf)# address-family ipv4</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> redistribute { { bgp</td>
<td>connected</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# redistribute bgp 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> route-policy route-policy-name { in</td>
<td>out }</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# route-policy policy_A in</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

**Command or Action**  
**default-metric** bandwidth delay reliability loading mtu  
**Example:**  
RP/0/RSP0/CP00:router(config-eigrp-vrf-af)# default-metric 1000 100 250 100 1500

### 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 ]`  
4. `log-neighbor-changes`  
5. `log-neighbor-warnings`  
6. `commit`  
7. `clear eigrp [ as-number ] [ vrf { vrf | all } ] [ ipv4 ] neighbors [ ip-address | type interface-path-id ]`  
8. `clear eigrp [ as-number ] [ vrf { vrf | all } ] [ ipv4 ] topology [ prefix mask ] [ prefix / length ]`  
9. `show eigrp [ as-number ] [ vrf { vrf | all } ] [ ipv4 ] accounting`  
10. `show eigrp [ as-number ] [ vrf { vrf | all } ] [ ipv4 ] interfaces [ type interface-path-id ] [ detail ]`  
11. `show eigrp [ as-number ] [ vrf { vrf | all } ] [ ipv4 ] neighbors [ detail ] [ type interface-path-id | static ]`  
12. `show protocols eigrp [ vrf vrf-name ]`  
13. `show eigrp [ as-number ] [ vrf { vrf | all } ] [ ipv4 ] topology [ ip-address mask ] [ active | all-links | detail-links | pending | summary | zero-successors ]`  
14. `show eigrp [ as-number ] [ vrf { vrf | all } ] [ ipv4 ] traffic`

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>Step 2</td>
<td><code>router eigrp as-number</code></td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config)# router eigrp 100</code></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>address-family [ ipv4 ]</code></td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-eigrp)# address-family ipv4</code></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>log-neighbor-changes</code></td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-eigrp-af)# log-neighbor-changes</code></td>
</tr>
<tr>
<td>Step 5</td>
<td><code>log-neighbor-warnings</code></td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-eigrp-af)# log-neighbor-warnings</code></td>
</tr>
<tr>
<td>Step 6</td>
<td><code>commit</code></td>
</tr>
<tr>
<td>Step 7</td>
<td>`clear eigrp [ as-number ] [ vrf { vrf</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router# clear eigrp 20 neighbors GigabitEthernet 0/1/0/0</code></td>
</tr>
<tr>
<td>Step 8</td>
<td>`clear eigrp [ as-number ] [ vrf { vrf</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router# clear eigrp topology</code></td>
</tr>
<tr>
<td>Step 9</td>
<td>`show eigrp [ as-number ] [ vrf { vrf</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router# show eigrp vrf all accounting</code></td>
</tr>
<tr>
<td>Step 10</td>
<td>`show eigrp [ as-number ] [ vrf { vrf</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router# show eigrp interfaces detail</code></td>
</tr>
</tbody>
</table>
### Configuring an EIGRP Authentication Keychain

Perform the following tasks to configure an authentication keychain on EIGRP interfaces.

#### Configuring an Authentication Keychain for an IPv4/IPv6 Interface on a Default VRF

Perform this task to configure an authentication keychain for an IPv4/IPv6 interface on a default VRF.

#### SUMMARY STEPS

1. configure
2. router eigrp  
   as-number
3. address-family  
   { ipv4  | ipv6 }
4. interface  
   type interface-path-id
5. authentication keychain  
   keychain-name
6. commit
DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router eigrp <em>as-number</em></td>
<td>Specifies the autonomous system number of the routing process to configure an EIGRP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router eigrp 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family { ipv4</td>
<td>ipv6 }</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-eigrp)# address-family ipv4</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> interface <em>type interface-path-id</em></td>
<td>Configures the interface on which EIGRP the routing protocol runs.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-eigrp-if)# interface gigabitEthernet 0/1/5/0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> authentication keychain <em>keychain-name</em></td>
<td>Authenticates all EIGRP protocol traffic on the interface, based on the MD5 algorithm.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-eigrp-if)# authentication keychain</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>

Configuring an Authentication Keychain for an IPv4/IPv6 Interface on a Nondefault VRF

Perform this task to configure an authentication keychain for an IPv4/IPv6 interface on a nondefault VRF.

SUMMARY STEPS

1. configure
2. router eigrp *as-number*
3. vrf *vrf-name*
4. address-family { ipv4 | ipv6 }
5. interface *type interface-path-id*
6. authentication keychain *keychain-name*
7. commit
## Detailed Steps

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><code>router eigrp as-number</code></td>
<td>Specifies the autonomous system number of the routing process to configure an EIGRP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>router eigrp 100</code></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>vrf vrf-name</code></td>
<td>Creates a VRF instance and enters VRF configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>vrf vrf1</code></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>`address-family { ipv4</td>
<td>ipv6 }`</td>
</tr>
<tr>
<td>Example:</td>
<td><code>address-family ipv4</code></td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td><code>interface type interface-path-id</code></td>
<td>Configures the interface on which EIGRP runs.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>interface gigabitEthernet 0/1/5/0</code></td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td><code>authentication keychain keychain-name</code></td>
<td>Authenticates all EIGRP protocol traffic on the interface, based on the MD5 algorithm.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>authentication keychain</code></td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td><code>commit</code></td>
<td></td>
</tr>
</tbody>
</table>

## 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
```
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.

```
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 PE-CE Configuration with Prefix-Limits: Example

The following example shows how to configure EIGRP to operate as a PE-CE protocol on a PE router. The configuration is under VRF CUSTOMER_1. A maximum prefix is typically configured to ensure that one set of customer routes do not overwhelm the EIGRP process.

```
router eigrp 500
  vrf CUSTOMER_1
    address-family ipv4
      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
```

Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x
**Configuring an EIGRP Authentication Keychain: Example**

The following example shows how to configure an authentication keychain for an IPv4 interface on a nondefault VRF:

```config
config
teacher eigrp 100
vrf vrf1
address-family ipv4
interface POS 0/1/0/0
authentication keychain key1
```

The following example shows how to configure an authentication keychain for an IPv6 interface on a default VRF:

```config
config
teacher eigrp 100
address-family ipv6
interface POS 0/1/0/0
authentication keychain key2
```

**Additional References**

The following sections provide references related to implementing EIGRP.

**Related Documents**

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIGRP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td>Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference</td>
</tr>
<tr>
<td>MPLS VPN support for EIGRP feature information</td>
<td>Implementing MPLS Layer 3 VPNs module and Implementing MPLS Layer 2 VPNs module in Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide</td>
</tr>
<tr>
<td>Site of Origin (SoO) support for EIGRP feature information</td>
<td>Implementing MPLS Traffic Engineering on Cisco ASR 9000 Series Router module in Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide</td>
</tr>
</tbody>
</table>
## Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>

## MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>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: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></td>
</tr>
</tbody>
</table>

## RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified RFCs are supported by this feature, and support for existing standards has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>

## Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a></td>
</tr>
</tbody>
</table>
Implementing IS-IS


This module describes how to implement IS-IS (IPv4 and IPv6) on your Cisco IOS XR network.

Currently, only default VRF is supported. VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families, L3VPN and Multicast will be supported in a future release.

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.

Restrictions for Implementing IS-IS

When multiple instances of IS-IS are being run, an interface can be associated with only one instance (process). Instances may not share an interface.
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.

Restrictions

When multiple instances of IS-IS are being run, an interface can be associated with only one instance (process). Instances may not share an interface.

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


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).
For information on IS-IS support for Bidirectional Forwarding Detection (BFD), see Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide and Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Command Reference.

### 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:

```plaintext
RP/0/RSP0/CPU0:router# configuration
RP/0/RSP0/CPU0:router(config)# router isis isp
RP/0/RSP0/CPU0:router(config-isis)#
```

#### Router Address Family Configuration Mode

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

```plaintext
RP/0/RSP0/CPU0:router(config)# router isis isp
RP/0/RSP0/CPU0:router(config-isis)# address-family
ipv4 u
nicast
RP/0/RSP0/CPU0:router(config-isis-af)#
```

#### Interface Configuration Mode

The following example shows how to enter interface configuration mode:

```plaintext
RP/0/RSP0/CPU0:router(config)# router isis isp
RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet 0/3/0/0
RP/0/RSP0/CPU0:router(config-isis-if)#
```
Interface Address Family Configuration Mode

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

```
RP/0/RSP0/CPU0:router(config)# router isis isp
RP/0/RSP0/CPU0:router(config-isis)# interface
GigabitEthernet 0 /3/0/0
RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-isis-if-af)#
```

IS-IS Interfaces

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

- **Active**—advertisies connected prefixes and forms adjacencies. This is the default for interfaces.
- **Passive**—advertisies 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 Cisco ASR 9000 Series Router module of *Cisco ASR 9000 Series Aggregation Services Router IP Addresses and Services Command Reference*. 
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 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, NSF minimizes the amount of time a network is unavailable to its users following a route processor (RP) failover. The main objective of NSF is to continue forwarding IP packets and perform a graceful restart following an RP failover.

When a router 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 in NSF-aware devices, 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 an RP failover. When the NSF feature is configured, peer networking devices do not experience routing flaps. Data traffic is forwarded through intelligent line cards while the standby RP assumes control from the failed active RP during a failover. The ability of line cards to remain up through a failover and to be kept current with the Forwarding Information Base (FIB) on the active RP is key to NSF operation.

When the Cisco IOS XR router running IS-IS routing performs an RP failover, the router must perform two tasks to resynchronize its link-state database with 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 neighbor routers are running a software version that supports the IETF Internet draft for router restartability, they assist an IETF NSF router that is restarting. With IETF NSF, 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.
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.

**Multi-Instance IS-IS**

You can configure up to eight 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 causes 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:

```bash
mpls traffic-eng path-selection ignore overload
```

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

```bash
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.
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).

**Multicast-Intact Feature**

The multicast-intact feature provides the ability to run multicast routing (PIM) when IGP shortcuts are configured and active on the router. Both OSPFv2 and IS-IS support the multicast-intact feature. MPLS TE and IP multicast coexistence is supported in Cisco IOS XR software by using the `mpls traffic-eng multicast-intact` IS-IS or OSPF router command.

You can enable multicast-intact in the IGP when multicast routing protocols (PIM) are configured and IGP shortcuts are configured on the router. IGP shortcuts are MPLS tunnels that are exposed to IGP. The IGPs route the IP traffic over these tunnels to destinations that are downstream from the egress router of the tunnel (from an SPF perspective). PIM cannot use IGP shortcuts for propagating PIM joins because reverse path forwarding (RPF) cannot work across a unidirectional tunnel.

When you enable multicast-intact on an IGP, the IGP publishes a parallel or alternate set of equal-cost next-hops for use by PIM. These next-hops are called mcast-intact next-hops. The mcast-intact next-hops have the following attributes:

- They are guaranteed not to contain any IGP shortcuts.
- They are not used for unicast routing but are used only by PIM to look up an IPv4 next-hop to a PIM source.
- They are not published to the FIB.
- When multicast-intact is enabled on an IGP, all IPv4 destinations that were learned through link-state advertisements are published with a set equal-cost mcast-intact next-hops to the RIB. This attribute applies even when the native next-hops have no IGP shortcuts.
- In IS-IS, the max-paths limit is applied by counting both the native and mcast-intact next-hops together. (In OSPFv2, the behavior is slightly different.)

**Multicast Topology Support Using IS-IS**

Multicast topology support allows for the configuration of IS-IS multicast topologies for IPv4 or IPv6 routing. IS-IS maintains a separate topology for multicast and runs a separate Shortest Path First (SPF) over the multicast topology. IS-IS multicast inserts routes from the IS-IS multicast topology into the multicast-unicast
Routing Information Base (muRIB) table in the RIB for the corresponding address family. Since PIM uses the muRIB, PIM uses routes from the multicast topology instead of routes from the unicast topology.

**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 (0xFFFFFFFF) is not considered for shortest path first (SPF). Therefore, the maximum wide metric of -1 (0xFFFFFFFF) 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 262 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-if-conv-frmkw-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 Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide for information on configuring MPLS IPFRR.
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.

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router isis instance-id</td>
<td>Enables the router to perform IS-IS routing for the specified routing instance.</td>
</tr>
</tbody>
</table>
| **Example:** | ```
RP/0/RSP0/CPU0:router(config)# router isis
``` |
| **Step 2** is-type { level-1 | level-1-2 | level-2-only } | Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode. |
| **Example:** | ```
RP/0/RSP0/CPU0:router(config-isis)# is-type level-1-2
``` |
| **Step 3** net network-entity-title | Configures network entity titles (NETs) for the routing instance. |
| **Example:** | ```
RP/0/RSP0/CPU0:router(config-isis)# net 47.0004.004d.0001.0001.0c11.1110.00
``` |
| **Step 4** show isis [ instance instance-id ] protocol | Displays the IS-IS protocol configuration. |
| **Example:** | ```
RP/0/RSP0/CPU0:router(config-isis)# show isis
``` |
### 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**

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 | multicast ]
13. commit
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters interface configuration mode.</td>
</tr>
<tr>
<td>Step 2 interface type interface-path-id</td>
<td>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.</td>
</tr>
</tbody>
</table>

Example: RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet 0/1/0/3

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  

or  

Specifies an IPv6 network assigned to the interface and enables IPv6 processing on the interface with the eui-64 keyword.  

or
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-if)# ipv4 address 10.0.1.3 255.255.255.0 &lt;br&gt;or &lt;br&gt;RP/0/RSP0/CPU0:router(config-if)# ipv6 address 3ffe:1234:c18:1::/64 eui-64 link-local &lt;br&gt;RP/0/RSP0/CPU0:router(config-if)# ipv6 address FE80::260:3EFF:FE11:6770 link-local &lt;br&gt;RP/0/RSP0/CPU0:router(config-if)# ipv6 enable &lt;br&gt;or</td>
<td>Specifies an IPv6 address assigned to the interface and enables IPv6 processing on the interface with the <code>link-local</code> keyword. &lt;br&gt;or &lt;br&gt;Automatically configures an IPv6 link-local address on the interface while also enabling the interface for IPv6 processing. &lt;br&gt;• The link-local address can be used only to communicate with nodes on the same link. &lt;br&gt;• Specifying the <code>ipv6 address ipv6-prefix / prefix-length</code> interface configuration command without the <code>eui-64</code> keyword configures site-local and global IPv6 addresses. &lt;br&gt;• Specifying the <code>ipv6 address ipv6-prefix / prefix-length</code> command with the <code>eui-64</code> 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. &lt;br&gt;• Specifying the <code>ipv6 address</code> command with the <code>link-local</code> keyword configures a link-local address on the interface that is used instead of the link-local address that is automatically configured when IPv6 is enabled on the interface.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>exit</strong></td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-if)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>router isis instance-id</strong></td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config)# router isis isp</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>net network-entity-title</strong></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td>address-family ipv6 [unicast]</td>
<td>Specifies the IPv6 address family and enters router address family</td>
</tr>
<tr>
<td></td>
<td>configuration mode.</td>
</tr>
<tr>
<td></td>
<td>• This example specifies the unicast IPv6 address family.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis)#</td>
<td></td>
</tr>
<tr>
<td>address-family ipv6 unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td>single-topology</td>
<td>(Optional) Configures the link topology for IPv4 when IPv6 is</td>
</tr>
<tr>
<td></td>
<td>configured.</td>
</tr>
<tr>
<td></td>
<td>• The <strong>single-topology</strong> command is valid only in IPv6 submode.</td>
</tr>
<tr>
<td></td>
<td>The command instructs IPv6 to use the single topology rather than the</td>
</tr>
<tr>
<td></td>
<td>default configuration of a separate topology in the multitopology</td>
</tr>
<tr>
<td></td>
<td>mode.</td>
</tr>
</tbody>
</table>
|                                                       | • See the Single-Topology IPv6 Support, on page 236 for more information.
| Example:                                              |                                                                         |
| RP/0/RSP0/CPU0:router(config-isis-af)#                |                                                                         |
| single-topology                                       |                                                                         |
| **Step 9**                                            | **Purpose**                                                             |
| exit                                                  | Exits router address family configuration mode, and returns the router  |
|                                                       | to router configuration mode.                                           |
| Example:                                              |                                                                         |
| RP/0/RSP0/CPU0:router(config-isis-af)#                |                                                                         |
| exit                                                  |                                                                         |
| **Step 10**                                           | **Purpose**                                                             |
| interface type interface-path-id                     | Enters interface configuration mode.                                    |
| Example:                                              |                                                                         |
| RP/0/RSP0/CPU0:router(config-isis)#                   |                                                                         |
| interface GigabitEthernet 0/1/0/3                     |                                                                         |
| **Step 11**                                           | **Purpose**                                                             |
| circuit-type {level-1 | level-1-2 | level-2-only }                                                      | (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.                         |
| Example:                                              |                                                                         |
| RP/0/RSP0/CPU0:router(config-isis-if)#                |                                                                         |
| circuit-type level-1-2                               |                                                                         |
| **Step 12**                                           | **Purpose**                                                             |
| address-family {ipv4 | ipv6 } [unicast | multicast ]                                                      | Specifies the IPv4 or IPv6 address family, and enters interface address|
|                                                       | family configuration mode.                                             |
|                                                       | • This example specifies the unicast IPv4 address family on the         |
| Example:                                              | interface.                                                             |
| RP/0/RSP0/CPU0:router(config-isis-if)#                |                                                                         |
| address-family ipv4 unicast                           |                                                                         |
| **Step 13**                                           | **Purpose**                                                             |
| commit                                                |                                                                         |
| **Step 14**                                           | **Purpose**                                                             |
| show isis [instance instance-id] interface [type     | (Optional) Displays information about the IS-IS interface.             |
| interface-path-id ] [detail] [level {1 |2} ]          |                                                                         |

---

**Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x**

248
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router# show isis interface GigabitEthernet 0/1/0/1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong> show isis [ instance instance-id ] topology [ systemid system-id ] [ level { 1</td>
<td>2 } ] [ summary ]</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router# show isis topology</td>
<td></td>
</tr>
</tbody>
</table>

### Configuring Multitopology Routing

This set of procedures configures multitopology routing, which is used by PIM for reverse-path forwarding (RPF) path selection.

### Restrictions for Configuring Multitopology Routing

- Only the default VRF is currently supported in a multitopology solution.
- Only protocol-independent multicast (PIM) and intermediate system-intermediate system (IS-IS) routing protocols are currently supported.
- Topology selection is restricted solely to (S, G) route sources for both SM and SSM. Static and IS-IS are the only interior gateway protocols (IGPs) that support multitopology deployment.

For non-(S, G) route sources like a rendezvous point or bootstrap router (BSR), or when a route policy is not configured, the current policy default remains in effect. In other words, either a unicast-default or multicast-default table is selected for all sources, based on OSPF/IS-IS/Multiprotocol Border Gateway Protocol (MBGP) configuration.

### Note

Although both **multicast** and **unicast** keywords are available when using the **address-family** [ipv4 | ipv6] command in routing policy language (RPL), only topologies under multicast SAFI can be configured globally.

### Information About Multitopology Routing

Configuring multitopology networks requires the following tasks:
Configuring a Global Topology and Associating It with an Interface

Follow these steps to enable a global topology in the default VRF and to enable its use with a specific interface.

### SUMMARY STEPS

1. `configure`
2. `address-family { ipv4 | ipv6 } multicast topology topo-name`
3. `maximum prefix limit`
4. `interface type interface-path-id`
5. `address-family { ipv4 | ipv6 } multicast topology topo-name`
6. Repeat Step 4 and Step 5 until you have specified all the interface instances you want to associate with your topologies.
7. `commit`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>configure</code></td>
<td>Configure a topology in the default VRF table that will be associated with an interface.</td>
</tr>
<tr>
<td>Step 2</td>
<td>`address-family { ipv4</td>
<td>ipv6 } multicast topology topo-name`</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# address-family ipv4 multicast topology green</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>maximum prefix limit</code></td>
<td>(Optional) Limits the number of prefixes allowed in a topology routing table. Range is 32 to 2000000.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-af)# maximum prefix 100</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>interface type interface-path-id</code></td>
<td>Specifies the interface to be associated with the previously specified VRF table that will add the connected and local routes to the appropriate routing table.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-af)# interface GigabitEthernet 0/3/0/0</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>`address-family { ipv4</td>
<td>ipv6 } multicast topology topo-name`</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-if)# address-family ipv4 multicast topology green</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>Repeat Step 4 and Step 5 until you have specified all the interface instances you want to associate with your topologies.</td>
<td></td>
</tr>
</tbody>
</table>
### Enabling an IS-IS Topology

To enable a topology in IS-IS, you must associate an IS-IS topology ID with the named topology. IS-IS uses the topology ID to differentiate topologies in the domain.

**Note**

This command must be configured prior to other topology commands.

### SUMMARY STEPS

1. `configure`
2. `router isis instance-id`
3. `address-family { ipv4 | ipv6 } multicast topology topo-name`
4. `topology-id multitopology-id`
5. `commit`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> <code>configure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>router isis instance-id</code></td>
<td>Enters IS-IS configuration submode.</td>
</tr>
<tr>
<td><strong>Step 3</strong> `address-family { ipv4</td>
<td>ipv6 } multicast topology topo-name`</td>
</tr>
</tbody>
</table>

```text
RP/0/RSP0/CPU0:router(config)# router isis purple

RP/0/RSP0/CPU0:router(config-isis)# address-family ipv4 multicast topology purple

RP/0/RSP0/CPU0:router(config-if-af)#
```
### Placing an Interface in a Topology in IS-IS

To associate an interface with a topology in IS-IS, follow these steps.

#### SUMMARY STEPS

1. `configure`
2. `router isis instance-id`
3. `net network-entity-title`
4. `interface type interface-path-id`
5. `address-family { ipv4 | ipv6 } multicast topology topo-name`
6. Repeat Step 4, on page 252 and Step 5, on page 253 until you have specified all the interface instances and associated topologies you want to configure in your network.
7. `commit`

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>configure</code></td>
<td>Enters IS-IS configuration submode.</td>
</tr>
<tr>
<td>2</td>
<td><code>router isis instance-id</code></td>
<td>Creates a network entity title for the configured isis interface.</td>
</tr>
<tr>
<td></td>
<td>Example: &lt;br&gt;RP/0/RSP0/CPU0:router(config)# router isis purple</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><code>net network-entity-title</code></td>
<td>Enters isis interface configuration submode and creates an interface instance.</td>
</tr>
<tr>
<td></td>
<td>Example: &lt;br&gt;RP/0/RSP0/CPU0:router(config-isis)# net netname</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><code>interface type interface-path-id</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: &lt;br&gt;RP/0/RSP0/CPU0:router(config-isis)# interface gigabitethernet 0/3/0/0</td>
<td></td>
</tr>
</tbody>
</table>
### Configuring IS-IS

<table>
<thead>
<tr>
<th>Step 5</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
|        | address-family \{ ipv4 | ipv6 \} multicast topology topo-name | - Enters isis address-family interface configuration submode.  
|        | Example: | - Places the interface instance into a topology. |
|        | RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 multicast topology green | |

**Step 6**

Repeat **Step 4, on page 252** and **Step 5, on page 253** until you have specified all the interface instances and associated topologies you want to configure in your network.

**Step 7**

commit

### Configuring a Routing Policy

For more information about creating a routing policy and about the `set rpf-topology` command, see [Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference](#).

**SUMMARY STEPS**

1. configure  
2. route-policy policy-name  
3. end-policy  
4. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>configure</td>
<td>Defines a routing policy and enters routing policy configuration submode.</td>
</tr>
</tbody>
</table>

**Step 2**

route-policy policy-name

**Example:**

```
RP/0/RSP0/CPU0:router(config)# route-policy mt1
RP/0/RSP0/CPU0:router(config-rpl)# if destination in 225.0.0.1, 225.0.0.11 then
RP/0/RSP0/CPU0:router(config-rpl-if)# if source in (10.10.10.10) then
  RP/0/RSP0/CPU0:router(config-rpl-if-2)# set rpf-topology ipv4 multicast topology greentable
  RP/0/RSP0/CPU0:router(config-rpl-if-2)# else
  RP/0/RSP0/CPU0:router(config-rpl-if-else-2)# set rpf-topology ipv4 multicast topology bluetable
  RP/0/RSP0/CPU0:router(config-rpl-if-else-2)# endif
RP/0/RSP0/CPU0:router(config-rpl-if)# endif
```

For detailed information about the use of the `set-rpf-topology` and other routing configuration commands, see [Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference](#).
### Configuring Multitopology for IS-IS

Multitopology is configured in the same way as the single topology. However, the `single-topology` command is omitted, invoking the default multitopology behavior. This task is optional.

### 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/RSP0/CPU0:router(config-isis)# lsp-refresh-interval 1200 level 2
RP/0/RSP0/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. **commit**
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>router isis instance-id</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config)# router isis lsp</td>
</tr>
<tr>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.&lt;br&gt;• You can change the level of routing to be performed by a particular routing instance by using the <strong>is-type</strong> router configuration command.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>**lsp-refresh-interval seconds [ level { 1</td>
</tr>
<tr>
<td>(Optional) Sets the time between regeneration of LSPs that contain different sequence numbers&lt;br&gt;• The refresh interval should always be set lower than the <strong>max-lsp-lifetime</strong> command.</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>lsp-check-interval</strong> `seconds [ level { 1</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-isis)# lsp-check-interval 240</code></td>
</tr>
<tr>
<td></td>
<td>(Optional) Configures the time between periodic checks of the entire database to validate the checksums of the LSPs in the database.</td>
</tr>
<tr>
<td></td>
<td>• This operation is costly in terms of CPU and so should be configured to occur infrequently.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>lsp-gen-interval</strong> <code>{ [ initial-wait </code>initial`</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-isis)# lsp-gen-interval maximum-wait 15 initial-wait 5</code></td>
</tr>
<tr>
<td></td>
<td>(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.</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>lsp-mtu</strong> `bytes [ level { 1</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-isis)# lsp-mtu 1300</code></td>
</tr>
<tr>
<td></td>
<td>(Optional) Sets the maximum transmission unit (MTU) size of LSPs.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>max-lsp-lifetime</strong> `seconds [ level { 1</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-isis)# max-lsp-lifetime 11000</code></td>
</tr>
<tr>
<td></td>
<td>(Optional) Sets the initial lifetime given to an LSP originated by the router.</td>
</tr>
<tr>
<td></td>
<td>• This is the amount of time that the LSP persists in the database of a neighbor unless the LSP is regenerated or refreshed.</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>ignore-lsp-errors disable</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-isis)# ignore-lsp-errors disable</code></td>
</tr>
<tr>
<td></td>
<td>(Optional) Sets the router to purge LSPs received with checksum errors.</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td><strong>interface</strong> <code>type interface-path-id</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3</code></td>
</tr>
<tr>
<td></td>
<td>Enters interface configuration mode.</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td><strong>lsp-interval</strong> `milliseconds [ level { 1</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-isis-if)# lsp-interval 100</code></td>
</tr>
<tr>
<td></td>
<td>(Optional) Configures the amount of time between each LSP sent on an interface.</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Step 11</td>
<td>csnp-interval  <em>seconds</em>  [ level { 1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if)# csnp-interval 30 level 1</td>
</tr>
<tr>
<td>Step 12</td>
<td>retransmit-interval  <em>seconds</em>  [ level { 1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if)# retransmit-interval 60</td>
</tr>
<tr>
<td>Step 13</td>
<td>retransmit-throttle-interval  <em>milliseconds</em>  [ level { 1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if)# retransmit-throttle-interval 1000</td>
</tr>
<tr>
<td>Step 14</td>
<td>mesh-group  *{ number</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if)# mesh-group blocked</td>
</tr>
<tr>
<td>Step 15</td>
<td>commit</td>
</tr>
<tr>
<td>Step 16</td>
<td>show isis  <em>interface</em>  *{ type interface-path-id</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router# show isis interface GigabitEthernet 0/1/0/1 brief</td>
</tr>
<tr>
<td>Step 17</td>
<td>show isis  *{ instance instance-id</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router# show isis database level 1</td>
</tr>
<tr>
<td>Step 18</td>
<td>show isis  *{ instance instance-id</td>
</tr>
</tbody>
</table>

Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x

OL-26048-03

257
### 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. commit
8. show running-config  [ command ]

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
</tr>
</tbody>
</table>

---

### Example:

```
RP/0/RSP0/CPU0:router# show isis lsp-log
```

### Example:

```
RP/0/RSP0/CPU0:router# show isis database-log
```

---

### Step 19:

**show isis database-log  [ level  \{ 1  | 2 \} ]**

(Optional) Display IS-IS database log information.

```
RP/0/RSP0/CPU0:router# show isis database-log
```
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>router isis instance-id</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td></td>
<td>• You can change the level of routing to be performed by a particular routing instance by using the <strong>is-type</strong> router configuration command.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>**nsf { cisco</td>
</tr>
<tr>
<td>Example:</td>
<td>Enables NSF on the next restart.</td>
</tr>
<tr>
<td></td>
<td>• Enter the <strong>cisco</strong> keyword to run IS-IS in heterogeneous networks that might not have adjacent NSF-aware networking devices.</td>
</tr>
<tr>
<td></td>
<td>• Enter the <strong>ietf</strong> keyword to enable IS-IS in homogeneous networks where all adjacent networking devices support IETF draft-based restartability.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>nsf interface-expires number</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>Configures the number of resends of an acknowledged NSF-restart acknowledgment.</td>
</tr>
<tr>
<td></td>
<td>• If the resend limit is reached during the NSF restart, the restart falls back to a cold restart.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>nsf interface-timer seconds</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>Configures the number of seconds to wait for each restart acknowledgment.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>nsf lifetime seconds</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>Configures the maximum route lifetime following an NSF restart.</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td>• Setting this value too high results in stale routes.</td>
</tr>
<tr>
<td></td>
<td>• Setting this value too low could result in routes purged too soon.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>commit</strong></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>show running-config [ command ]</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>(Optional) Displays the entire contents of the currently running configuration file or a subset of that file.</td>
</tr>
<tr>
<td></td>
<td>• Verify that &quot;nsf&quot; appears in the IS-IS configuration of the NSF-aware device.</td>
</tr>
<tr>
<td></td>
<td>• This example shows the contents of the configuration file for the &quot;isp&quot; instance only.</td>
</tr>
</tbody>
</table>
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. `commit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>router isis instance-id</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config)# router isis isp</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>`lsp-password { hmac-md5</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-isis)# lsp-password hmac-md5 clear password1 level 1</td>
</tr>
<tr>
<td></td>
<td><strong>Step 4</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Step 5</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>they are sent. It does not authenticate received LSPs or SNPs.</td>
<td></td>
</tr>
<tr>
<td>• The <strong>snp send-only</strong> keyword adds authentication to SNPs when they are sent. It does not authenticate received SNPs.</td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong> To disable SNP password checking, the <strong>snp send-only</strong> keywords must be specified in the <strong>lsp-password</strong> command.</td>
<td></td>
</tr>
</tbody>
</table>

**Step 4**  
**interface type interface-path-id**  
**Example:**  
RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3

**Step 5**  
**hello-password { hmac-md5 | text } { clear | encrypted } password [ level { 1 | 2 } | send-only ]**  
**Example:**  
RP/0/RSP0/CPU0:router(config-isis-if)#hello-password text clear mypassword

**Step 6**  
**commit**

---

**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. lsp-password keychain keychain-name [ level { 1 | 2 } | send-only ] [ snp send-only ]
4. interface type interface-path-id
5. hello-password keychain keychain-name [ level { 1 | 2 } | send-only ]
6. commit
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Step 2</td>
<td><code>router isis instance-id</code></td>
<td>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 <code>is-type</code> command.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# router isis isp</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>`lsp-password keychain keychain-name [ level { 1</td>
<td>2 } ] [ send-only ] [ snp send-only ]`</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-isis)# lsp-password keychain isis_a level 1</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>interface type interface-path-id</code></td>
<td>Enters interface configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>`hello-password keychain keychain-name [ level { 1</td>
<td>2 } ] [ send-only ]`</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-isis-if)# hello-password keychain isis_b</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

## 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 Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide*

**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.
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. **commit**
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router isis instance-id</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# router isis isp</td>
<td>You can change the level of routing to be performed by a particular routing instance by using the is-type router configuration command.</td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family { ipv4</td>
<td>ipv6 } [ unicast ]</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-isis)#address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> mpls traffic-eng level { 1</td>
<td>2 }</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-isis-af)# mpls traffic-eng level 1</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>mpls traffic-eng router-id</strong> { **ip-address</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-isis-af)# mpls traffic-eng router-id loopback0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>metric-style</strong> { <strong>level</strong> { 1</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-isis-af)# metric-style wide level 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>commit</strong>&lt;br&gt;<strong>Step 8</strong></td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td><strong>show isis</strong> { <strong>instance</strong> instance-id } <strong>mpls traffic-eng</strong> adjacency-log&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router# show isis instance isp mpls traffic-eng adjacency-log&lt;br&gt;<strong>Optional</strong> Displays a log of MPLS TE IS-IS adjacency changes.</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td><strong>show isis</strong> { <strong>instance</strong> instance-id } <strong>mpls traffic-eng</strong> advertisements&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router# show isis instance isp mpls traffic-eng advertisements&lt;br&gt;<strong>Optional</strong> Displays the latest flooded record from MPLS TE.</td>
</tr>
</tbody>
</table>

---

**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. `hello-password { hmac-md5 | text } { clear | encrypted } password [ level { 1 | 2 } ] [ send-only ]`
9. `commit`
10. `show isis [ instance instance-id ] adjacency type 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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>configure</code></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>router isis instance-id</code></td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/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/RSP0/CPU0:router(config-isis)# log adjacency changes
```

Generates a log message when an IS-IS adjacency changes state (up or down).
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 4</td>
<td><strong>interface type interface-path-id</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3</td>
</tr>
<tr>
<td></td>
<td>Enters interface configuration mode.</td>
</tr>
<tr>
<td>Step 5</td>
<td><strong>hello-padding</strong> { disable</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if)# hello-padding sometimes</td>
</tr>
<tr>
<td></td>
<td>Configures padding on IS-IS hello PDUs for an IS-IS interface on the router.</td>
</tr>
<tr>
<td></td>
<td>• Hello padding applies to only this interface and not to all interfaces.</td>
</tr>
<tr>
<td>Step 6</td>
<td><strong>hello-interval seconds</strong> [ level { 1</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if)#hello-interval 6</td>
</tr>
<tr>
<td></td>
<td>Specifies the length of time between hello packets that the software sends.</td>
</tr>
<tr>
<td>Step 7</td>
<td><strong>hello-multiplier multiplier</strong> [ level { 1</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if)# hello-multiplier 10</td>
</tr>
<tr>
<td></td>
<td>Specifies the number of IS-IS hello packets a neighbor must miss before the router should declare the adjacency as down.</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td>• Conversely, not detecting the failure of an adjacent router can result in greater packet loss.</td>
</tr>
<tr>
<td>Step 8</td>
<td><strong>hello-password</strong> { hmac-md5</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if)# hello-password text clear mypassword</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Step 9</td>
<td><strong>commit</strong></td>
</tr>
<tr>
<td>Step 10</td>
<td><strong>show isis</strong> [ instance instance-id ] [ adjacency type interface-path-id ] [ detail ] [ systemid system-id ]</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router# show isis instance isp adjacency</td>
</tr>
<tr>
<td></td>
<td>(Optional) Displays IS-IS adjacencies.</td>
</tr>
<tr>
<td>Step 11</td>
<td><strong>show isis adjacency-log</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router# show isis adjacency-log</td>
</tr>
<tr>
<td></td>
<td>(Optional) Displays a log of the most recent adjacency state transitions.</td>
</tr>
</tbody>
</table>
### 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.

The incremental SPF algorithm can be enabled separately. When enabled, the incremental shortest path first (ISPF) is not employed immediately. Instead, the full SPF algorithm is used to “seed” the state information required for the ISPF to run. The startup delay prevents the ISPF from running for a specified interval after an IS-IS restart (to permit the database to stabilize). After the startup delay elapses, the ISPF is principally responsible for performing all of the SPF calculations. The reseed interval enables a periodic running of the full SPF to ensure that the ISFP state remains synchronized.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 12</strong> show isis [ instance instance-id ] interface [ type interface-path-id ] [ brief</td>
<td>detail ] [ level { 1</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router# show isis interface GigabitEthernet 0/1/0/1 brief</td>
</tr>
<tr>
<td><strong>Step 13</strong> show isis [ instance instance-id ] neighbors [ interface-type interface-instance ] [ summary</td>
<td>detail ] [ systemid system-id ]</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router# show isis neighbors summary</td>
</tr>
</tbody>
</table>
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. ispf [ level { 1 | 2 }]
6. commit
7. show isis [ instance instance-id ] [ [ ipv4 | ipv6 | afi-all ] [ unicast | safi-all ] ] spf-log [ level { 1 | 2 } ] [ ispf | fspf | prc | nhc ] [ detail | verbose ] [ last number | first number ]

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Step 2 router isis instance-id</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# router isis isp</td>
<td>You can change the level of routing to be performed by a particular routing instance by using the is-type router configuration command.</td>
</tr>
<tr>
<td>Step 3 address-family { ipv4</td>
<td>ipv6 } [ unicast ]</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-isis)#address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td>Step 4 spf-interval { [ initial-wait initial</td>
<td>secondary-wait secondary</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-isis-af)# spf-interval initial-wait 10 maximum-wait 30</td>
<td>• This value imposes a delay in the SPF computation after an event trigger and enforces a minimum elapsed time between SPF runs.</td>
</tr>
<tr>
<td></td>
<td>• If this value is configured too low, the router can lose too many CPU resources when the network is unstable.</td>
</tr>
<tr>
<td></td>
<td>• Configuring the value too high delays changes in the network topology that result in lost packets.</td>
</tr>
<tr>
<td></td>
<td>• The SPF interval does not apply to the running of the ISPF because that algorithm runs immediately on receiving a changed LSP.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>ispf [ level { 1</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-isis-af)# ispf</td>
</tr>
<tr>
<td></td>
<td>(Optional) Configures incremental IS-IS ISPF to calculate network topology.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>commit</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>show isis [ instance instance-id ][ ipv4</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router# show isis instance 1 spf-log ipv4</td>
</tr>
<tr>
<td></td>
<td>(Optional) Displays how often and why the router has run a full SPF calculation.</td>
</tr>
</tbody>
</table>

## 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. set-attached-bit
11. commit
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables IS-IS routing for the specified routing process, and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router isis instance-id</td>
<td>Enables IS-IS routing for the specified routing process, and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router isis isp</td>
</tr>
<tr>
<td><strong>Step 3</strong> set-overload-bit [ on-startup { delay</td>
<td>wait-for-bgp }] [ level { 1</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis)# set-overload-bit</td>
</tr>
<tr>
<td><strong>Step 4</strong> address-family { ipv4</td>
<td>ipv6 } [ unicast ]</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis)# address-family ipv4 unicast</td>
</tr>
<tr>
<td><strong>Step 5</strong> default-information originate [ route-policy route-policy-name ]</td>
<td>(Optional) Injects a default IPv4 or IPv6 route into an IS-IS routing domain.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-af)# default-information originate</td>
</tr>
<tr>
<td><strong>Step 6</strong> redistribute isis instance [ level-1</td>
<td>level-2</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-af)# redistribute isis 2 level-1</td>
</tr>
<tr>
<td><strong>Step 7</strong> Do one of the following:</td>
<td>(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.</td>
</tr>
<tr>
<td>• summary-prefix address / prefix-length [ level { 1</td>
<td>2 }]</td>
</tr>
<tr>
<td>• summary-prefix ipv6-prefix / prefix-length [ level { 1</td>
<td>2 }]</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
</tbody>
</table>
| RP/0/RSP0/CPU0:router(config-isis-af)# summary-prefix 10.1.0.0/16 level 1 or RP/0/RSP0/CPU0:router(config-isis-af)# summary-prefix 3003:xxxx::/24 level 1 | • 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.  
• 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*  
((Optional) Configures the maximum number of parallel paths allowed in a routing table.)

**Example:**  
RP/0/RSP0/CPU0:router(config-isis-af)# maximum-paths 16

**Step 9**  
**distance**  
*weight*  
*[address / prefix-length]*  
*[route-list-name]*]  
(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.

**Example:**  
RP/0/RSP0/CPU0:router(config-isis-af)# distance 90

**Step 10**  
**set-attached-bit**  
(Optional) Configures an IS-IS instance with an attached bit in the Level 1 LSP.

**Example:**  
RP/0/RSP0/CPU0:router(config-isis-af)# set-attached-bit

**Step 11**  
**commit**

---

### 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  
3. instance-id  
4. interface  
5. type interface-path-id  
6. address-family  
7. ipv4  
8. unicast  
9. mpls ldp sync [ level { 1 | 2 } ]  
10. commit
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
</tr>
<tr>
<td>Step 2</td>
<td>router isis instance-id</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router isis isp</td>
</tr>
<tr>
<td></td>
<td>Enables IS-IS routing for the specified routing process, and places the router in router configuration mode.</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td>Step 3</td>
<td>interface type interface-path-id</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3</td>
</tr>
<tr>
<td></td>
<td>Enters interface configuration mode.</td>
</tr>
<tr>
<td>Step 4</td>
<td>address-family ipv4 unicast</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast</td>
</tr>
<tr>
<td></td>
<td>Specifies the IPv4 address family and enters router address family configuration mode.</td>
</tr>
<tr>
<td>Step 5</td>
<td>mpls ldp sync [ level { 1</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)# mpls ldp sync level 1</td>
</tr>
<tr>
<td></td>
<td>Enables MPLS LDP synchronization for the IPv4 address family under interface GigabitEthernet 0/1/0/3.</td>
</tr>
<tr>
<td>Step 6</td>
<td>commit</td>
</tr>
</tbody>
</table>

## Enabling Multicast-Intact

This optional task describes how to enable multicast-intact for IS-IS routes that use IPv4 and IPv6 addresses.

### SUMMARY STEPS

1. configure  
2. `router isis instance-id`  
3. `address-family { ipv4 | ipv6 } [ unicast | multicast ]`  
4. `mpls traffic-eng multicast-intact`  
5. commit
DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router isis instance-id</td>
<td>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.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router isis isp</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family { ipv4</td>
<td>ipv6 } [ unicast</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> mpls traffic-eng multicast-intact</td>
<td>Enables multicast-intact.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis-af)# mpls traffic-eng multicast-intact</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>

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. commit
10. show isis [ ipv4 | ipv6 | afi-all ] [ unicast | safi-all ] route [ detail ]
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
</tbody>
</table>
| **Step 2** | router isis instance-id  
Example:  
RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-isis-af)# exit | Exits router address family configuration mode, and returns the router to router configuration mode. |
| **Step 6** | interface type number  
Example:  
RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3 | Enters interface configuration mode. |
| **Step 7** | address-family (ipv4 | ipv6 ) [ unicast ]  
Example:  
RP/0/RSP0/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/RSP0/CPU0:router(config-isis-if-af)# tag 3 | Sets the value of the tag to associate with the advertised connected route. |
| **Step 9** | commit | |
### 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. `commit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>configure</code></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>router isis instance-id</code></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config)# router isis isp</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>`address-family { ipv4</td>
<td>ipv6 } [ unicast ]`</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-isis)# address-family ipv4 unicast</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>`metric-style wide [ transition ] [ level { 1</td>
<td>2 } ]`</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RSP0/CPU0:router(config-isis-af)# metric-style wide level 1</code></td>
<td></td>
</tr>
</tbody>
</table>
Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 5</strong></td>
<td>installs all routes tagged with the value 3 first.</td>
</tr>
<tr>
<td>spf prefix-priority { level { 1</td>
<td>2 }</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Step 6</strong> commit</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis-af)# spf prefix-priority high tag 3</td>
<td></td>
</tr>
</tbody>
</table>

**Configuring IP/LDP Fast Reroute**

This optional task describes how to enable the IP/LDP fast reroute computation to converge traffic flows around link failures.

**Note**

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

**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 | ipv6 } [ unicast ]
6. fast-reroute { per-link | per-prefix }
7. Do one of the following:
   - fast-reroute per-link { level { 1 | 2 } }
   - fast-reroute per-prefix { level { 1 | 2 } }
8. Do one of the following:
   - fast-reroute per-link exclude interface type interface-path-id { level { 1 | 2 } }
   - fast-reroute per-prefix exclude interface type interface-path-id { level { 1 | 2 } }
9. Do one of the following:
   - fast-reroute per-link lfa-candidate interface type interface-path-id { level { 1 | 2 } }
   - fast-reroute per-prefix lfa-candidate interface type interface-path-id { level { 1 | 2 } }
10. commit
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router isis <em>instance-id</em></td>
<td>Enters interface configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router isis isp</td>
</tr>
<tr>
<td><strong>Step 3</strong> interface <em>type interface-path-id</em></td>
<td>(Optional) Configures the type of adjacency.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3</td>
</tr>
<tr>
<td><strong>Step 4</strong> circuit-type { level-1</td>
<td>level-1-2</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router8(config-isis-if)# circuit-type level-1</td>
</tr>
<tr>
<td><strong>Step 5</strong> address-family { ipv4</td>
<td>ipv6 } [ unicast ]</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)# fast-reroute per-link</td>
</tr>
<tr>
<td><strong>Step 6</strong> fast-reroute { per-link</td>
<td>per-prefix }</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)#fast-reroute per-link level 1</td>
</tr>
<tr>
<td><strong>Step 7</strong> Do one of the following:</td>
<td>• per-prefix—Used for prefix dependent computation.</td>
</tr>
<tr>
<td>• fast-reroute per-link { level { 1</td>
<td>2 } }</td>
</tr>
<tr>
<td>• fast-reroute per-prefix { level { 1</td>
<td>2 } }</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)#fast-reroute per-link level 1</td>
</tr>
<tr>
<td></td>
<td>Or</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)#fast-reroute per-prefix level 2</td>
</tr>
</tbody>
</table>
### 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

---

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 8</strong></td>
<td>Do one of the following:</td>
</tr>
<tr>
<td></td>
<td>- `fast-reroute per-link exclude interface type interface-path-id { level { 1</td>
</tr>
<tr>
<td></td>
<td>- `fast-reroute per-prefix exclude interface type interface-path-id { level { 1</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)#fast-reroute per-link exclude interface Loopback0 level 1 Or</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)#fast-reroute per-prefix exclude interface POS0/6/0/0 level 2</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>Do one of the following:</td>
</tr>
<tr>
<td></td>
<td>- `fast-reroute per-link lfa-candidate interface type interface-path-id { level { 1</td>
</tr>
<tr>
<td></td>
<td>- `fast-reroute per-prefix lfa-candidate interface type interface-path-id { level { 1</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)#fast-reroute per-link lfa-candidate interface MgmtEth0/RP0/CPU0/0 level 1 Or</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)#fast-reroute per-prefix lfa-candidate interface MgmtEth0/RP1/CPU0/0 level 2</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>commit</td>
</tr>
</tbody>
</table>

### Implementing IS-IS

---

**Purpose**

Excludes an interface from fast-reroute computation.

**Step 8**

Do one of the following:

- `fast-reroute per-link exclude interface type interface-path-id { level { 1 | 2 } }`
- `fast-reroute per-prefix exclude interface type interface-path-id { level { 1 | 2 } }

**Example:**

RP/0/RSP0/CPU0:router(config-isis-if-af)#fast-reroute per-link exclude interface Loopback0 level 1
Or
RP/0/RSP0/CPU0:router(config-isis-if-af)#fast-reroute per-prefix exclude interface POS0/6/0/0 level 2

**Purpose**

Configures to include an interface to LFA candidate in fast-reroute computation.

**Step 9**

Do one of the following:

- `fast-reroute per-link lfa-candidate interface type interface-path-id { level { 1 | 2 } }`
- `fast-reroute per-prefix lfa-candidate interface type interface-path-id { level { 1 | 2 } }

**Example:**

RP/0/RSP0/CPU0:router(config-isis-if-af)#fast-reroute per-link lfa-candidate interface MgmtEth0/RP0/CPU0/0 level 1
Or
RP/0/RSP0/CPU0:router(config-isis-if-af)#fast-reroute per-prefix lfa-candidate interface MgmtEth0/RP1/CPU0/0 level 2

**Step 10**

commit
SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 mpls traffic-eng path-selection ignore overload</td>
<td>Activates IS-IS overload bit avoidance.</td>
</tr>
</tbody>
</table>

Example:
RP/0/RSP0/CPU0:router(config)# mpls traffic-eng path-selection ignore overload

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. `commit`
5. `show isis interface`
6. `show isis lsp`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>configure</code></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>router isis instance-id</code>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config)# router isis purple</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>`link-group link-group-name { [ metric-offset count</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><code>commit</code></td>
</tr>
<tr>
<td>Step 5</td>
<td>Command or Action</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>show isis interface</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router# show isis interface</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 6</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>show isis lsp</td>
<td>(Optional) Displays the updated metric value.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router# show isis lsp</td>
<td></td>
</tr>
</tbody>
</table>

**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
  FSNP Entry Queue Size: 0
CLNS I/O
  Protocol State: Up
  MTU: 1497
  SNPA: 0026.9829.af19
Layer-2 MCast Groups Membership:
  All ISs: Yes
```
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. commit
7. show isis interface

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router isis instance-id</td>
<td>Enters IS-IS configuration submode.</td>
</tr>
</tbody>
</table>

Example:
```
RP/0/RSP0/CPU0:router(config)# router isis purple
```
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 3</strong></td>
<td>interface  type  interface-path-id</td>
</tr>
<tr>
<td>Example:</td>
<td>Enters interface configuration mode.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet 0/1/0/3</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>address-family  ipv4</td>
</tr>
<tr>
<td>Example:</td>
<td>Specifies the IPv6 address family and enters router address family configuration mode.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>link-group  link-group-name</td>
</tr>
<tr>
<td>Example:</td>
<td>Specifies the link-group name and sets the tag at the level specified.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast link-group access level 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>commit</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>show isis interface</td>
</tr>
<tr>
<td>Example:</td>
<td>(Optional) If link-group is configured on the interface, when showing the IS-IS interface-related topology, this command displays the link-group value.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router# show isis interface</td>
<td></td>
</tr>
</tbody>
</table>

## 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
```
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 `set-attached-bit` 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
    set
    -attached
    -bit

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 *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide*:

- Implementing OSPF
- Implementing BGP
- Implementing EIGRP
- Implementing RIP
Additional References

The following sections provide references related to implementing IS-IS.

**Related Documents**

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS-IS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td>Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference</td>
</tr>
<tr>
<td>MPLS TE feature information</td>
<td>Implementing MPLS Traffic Engineering on Cisco ASR 9000 Series Router module in Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide</td>
</tr>
<tr>
<td>Bidirectional Forwarding Detection (BFD)</td>
<td>Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide and Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Command Reference</td>
</tr>
</tbody>
</table>

**Standards**

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft-ietf-isis-ipv6-05.txt</td>
<td>Routing IPv6 with IS-IS, by Christian E. Hopps</td>
</tr>
<tr>
<td>Draft-ietf-isis-traffic-05.txt</td>
<td>IS-IS Extensions for Traffic Engineering, by Henk Smit and Toni Li</td>
</tr>
<tr>
<td>Draft-ietf-isis-restart-04.txt</td>
<td>Restart Signaling for IS-IS, by M. Shand and Les Ginsberg</td>
</tr>
<tr>
<td>Draft-ietf-isis-igp-p2p-over-lan-05.txt</td>
<td>Point-to-point operation over LAN in link-state routing protocols, by Naiming Shen</td>
</tr>
</tbody>
</table>
### Standards

| Draft-ietf-rtgwg-lf-conv-frwk-00.txt | A Framework for Loop-free Convergence, by M. Shand and S. Bryant |

### MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>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: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></td>
</tr>
</tbody>
</table>

### RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 1142</td>
<td>OSI IS-IS Intra-domain Routing Protocol</td>
</tr>
<tr>
<td>RFC 1195</td>
<td>Use of OSI IS-IS for Routing in TCP/IP and Dual Environments</td>
</tr>
<tr>
<td>RFC 2763</td>
<td>Dynamic Hostname Exchange Mechanism for IS-IS</td>
</tr>
<tr>
<td>RFC 2966</td>
<td>Domain-wide Prefix Distribution with Two-Level IS-IS</td>
</tr>
<tr>
<td>RFC 2973</td>
<td>IS-IS Mesh Groups</td>
</tr>
<tr>
<td>RFC 3277</td>
<td>IS-IS Transient Blackhole Avoidance</td>
</tr>
<tr>
<td>RFC 3373</td>
<td>Three-Way Handshake for IS-IS Point-to-Point Adjacencies</td>
</tr>
<tr>
<td>RFC 3567</td>
<td>IS-IS Cryptographic Authentication</td>
</tr>
<tr>
<td>RFC 4444</td>
<td>IS-IS Management Information Base</td>
</tr>
</tbody>
</table>
## Technical Assistance

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th><strong>Link</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>technical content, including links to products, technologies, solutions,</td>
<td></td>
</tr>
<tr>
<td>technical tips, and tools. Registered Cisco.com users can log in from this</td>
<td></td>
</tr>
<tr>
<td>page to access even more content.</td>
<td></td>
</tr>
</tbody>
</table>
Implementing OSPF

Open Shortest Path First (OSPF) is an Interior Gateway Protocol (IGP) developed by the OSPF working group of the Internet Engineering Task Force (IETF). Designed expressly for IP networks, OSPF supports IP subnetting and tagging of externally derived routing information. OSPF also allows packet authentication and uses IP multicast when sending and receiving packets.

OSPF Version 3 (OSPFv3) expands on OSPF Version 2, providing support for IPv6 routing prefixes.

This module describes the concepts and tasks you need to implement both versions of OSPF on your Cisco ASR 9000 Series Router. The term "OSPF" implies both versions of the routing protocol, unless otherwise noted.

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 377 section of this module. To locate documentation for other commands that might appear during execution of a configuration task, search online in the Cisco ASR 9000 Series Aggregation Services Router Commands Master List.

Feature History for Implementing OSPF

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.7.2</td>
<td>This feature was introduced.</td>
</tr>
<tr>
<td>Release 3.9.0</td>
<td>Support was added for the following features:</td>
</tr>
<tr>
<td></td>
<td>• OSPFv2 SPF Prefix Prioritization.</td>
</tr>
<tr>
<td></td>
<td>• IP fast reroute loop-free alternates computation</td>
</tr>
<tr>
<td></td>
<td>• Warm Standby for OSPF Version 3</td>
</tr>
<tr>
<td>Release 4.2.0</td>
<td>Support was added for the following features:</td>
</tr>
<tr>
<td></td>
<td>• OSPFv2 Fast Re-route Per-Prefix Computation</td>
</tr>
<tr>
<td></td>
<td>• OSPFv3 Non-stop Routing (NSR)</td>
</tr>
</tbody>
</table>
### Prerequisites for Implementing OSPF

The following are prerequisites for implementing OSPF on Cisco IOS XR software:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

- Configuration tasks for OSPFv3 assume that you are familiar with IPv6 addressing and basic configuration. See the Implementing Network Stack IPv4 and IPv6 on Cisco ASR 9000 Series Router module of the Cisco ASR 9000 Series Aggregation Services Router IP Addresses and Services Configuration Guide 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.

---

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 5.3.0</td>
<td>Support was added for the following features:</td>
</tr>
<tr>
<td></td>
<td>• OSPFv2 Segment Routing Topology Independent Fast Reroute</td>
</tr>
<tr>
<td></td>
<td>• 64 ECMP for ASR 9000 Enhanced Ethernet Line Card</td>
</tr>
<tr>
<td>Release 6.0.1</td>
<td>The following features were supported:</td>
</tr>
<tr>
<td></td>
<td>• Excessive Punt Flow Trap Processing</td>
</tr>
</tbody>
</table>
Information About Implementing OSPF

To implement OSPF you need to understand the following concepts:

**OSPF Functional Overview**

OSPF is a routing protocol for IP. It is a link-state protocol, as opposed to a distance-vector protocol. A link-state protocol makes its routing decisions based on the states of the links that connect source and destination machines. The state of the link is a description of that interface and its relationship to its neighboring networking devices. The interface information includes the IP address of the interface, network mask, type of network to which it is connected, routers connected to that network, and so on. This information is propagated in various types of link-state advertisements (LSAs).

A router stores the collection of received LSA data in a link-state database. This database includes LSA data for the links of the router. The contents of the database, when subjected to the Dijkstra algorithm, extract data to create an OSPF routing table. The difference between the database and the routing table is that the database contains a complete collection of raw data; the routing table contains a list of shortest paths to known destinations through specific router interface ports.

OSPF is the IGP of choice because it scales to large networks. It uses areas to partition the network into more manageable sizes and to introduce hierarchy in the network. A router is attached to one or more areas in a network. All of the networking devices in an area maintain the same complete database information about the link states in their area only. They do not know about all link states in the network. The agreement of the database information among the routers in the area is called convergence.

At the intradomain level, OSPF can import routes learned using Intermediate System-to-Intermediate System (IS-IS). OSPF routes can also be exported into IS-IS. At the interdomain level, OSPF can import routes learned using Border Gateway Protocol (BGP). OSPF routes can be exported into BGP.

Unlike Routing Information Protocol (RIP), OSPF does not provide periodic routing updates. On becoming neighbors, OSPF routers establish an adjacency by exchanging and synchronizing their databases. After that, only changed routing information is propagated. Every router in an area advertises the costs and states of its links, sending this information in an LSA. This state information is sent to all OSPF neighbors one hop away. All the OSPF neighbors, in turn, send the state information unchanged. This flooding process continues until all devices in the area have the same link-state database.

To determine the best route to a destination, the software sums all of the costs of the links in a route to a destination. After each router has received routing information from the other networking devices, it runs the shortest path first (SPF) algorithm to calculate the best path to each destination network in the database.

The networking devices running OSPF detect topological changes in the network, flood link-state updates to neighbors, and quickly converge on a new view of the topology. Each OSPF router in the network soon has the same topological view again. OSPF allows multiple equal-cost paths to the same destination. Since all link-state information is flooded and used in the SPF calculation, multiple equal cost paths can be computed and used for routing.

On broadcast and nonbroadcast multiaccess (NBMA) networks, the designated router (DR) or backup DR performs the LSA flooding. On point-to-point networks, flooding simply exits an interface directly to a neighbor.

OSPF runs directly on top of IP; it does not use TCP or User Datagram Protocol (UDP). OSPF performs its own error correction by means of checksums in its packet header and LSAs.
In OSPFv3, the fundamental concepts are the same as OSPF Version 2, except that support is added for the increased address size of IPv6. New LSA types are created to carry IPv6 addresses and prefixes, and the protocol runs on an individual link basis rather than on an individual IP-subnet basis.

OSPF typically requires coordination among many internal routers: Area Border Routers (ABRs), which are routers attached to multiple areas, and Autonomous System Border Routers (ASBRs) that export reroutes from other sources (for example, IS-IS, BGP, or static routes) into the OSPF topology. At a minimum, OSPF-based routers or access servers can be configured with all default parameter values, no authentication, and interfaces assigned to areas. If you intend to customize your environment, you must ensure coordinated configurations of all routers.

### Key Features Supported in the Cisco IOS XR Software OSPF Implementation

The Cisco IOS XR Software implementation of OSPF conforms to the OSPF Version 2 and OSPF Version 3 specifications detailed in the Internet RFC 2328 and RFC 2740, respectively.

The following key features are supported in the Cisco IOS XR Software implementation:

- **Hierarchy**—CLI hierarchy is supported.
- **Inheritance**—CLI inheritance is supported.
- **Stub areas**—Definition of stub areas is supported.
- **NSF**—Nonstop forwarding is supported.
- **SPF throttling**—Shortest path first throttling feature is supported.
- **LSA throttling**—LSA throttling feature is supported.
- **Fast convergence**—SPF and LSA throttle timers are set, configuring fast convergence. The OSPF LSA throttling feature provides a dynamic mechanism to slow down LSA updates in OSPF during network instability. LSA throttling also allows faster OSPF convergence by providing LSA rate limiting in milliseconds.
- **Route redistribution**—Routes learned using any IP routing protocol can be redistributed into any other IP routing protocol.
- **Authentication**—Plain text and MD5 authentication among neighboring routers within an area is supported.
- **Routing interface parameters**—Configurable parameters supported include interface output cost, retransmission interval, interface transmit delay, router priority, router "dead" and hello intervals, and authentication key.
- **Virtual links**—Virtual links are supported.
- **Not-so-stubby area (NSSA)**—RFC 1587 is supported.
- **OSPF over demand circuit**—RFC 1793 is supported.

### Comparison of Cisco IOS XR Software OSPFv3 and OSPFv2

Much of the OSPFv3 protocol is the same as in OSPFv2. OSPFv3 is described in RFC 2740.

The key differences between the Cisco IOS XR Software OSPFv3 and OSPFv2 protocols are as follows:
• **OSPFv3** expands on **OSPFv2** to provide support for IPv6 routing prefixes and the larger size of IPv6 addresses.

• When using an NBMA interface in **OSPFv3**, users must manually configure the router with the list of neighbors. Neighboring routers are identified by the link local address of the attached interface of the neighbor.

• Unlike in **OSPFv2**, multiple **OSPFv3** processes can be run on a link.

• LSAs in **OSPFv3** are expressed as "prefix and prefix length" instead of "address and mask."

• The router ID is a 32-bit number with no relationship to an IPv6 address.

### **OSPF Hierarchical CLI and CLI Inheritance**

**Cisco IOS XR Software** introduces new **OSPF configuration fundamentals** consisting of hierarchical CLI and CLI inheritance.

Hierarchical CLI is the grouping of related network component information at defined hierarchical levels such as at the router, area, and interface levels. Hierarchical CLI allows for easier configuration, maintenance, and troubleshooting of **OSPF** configurations. When configuration commands are displayed together in their hierarchical context, visual inspections are simplified. Hierarchical CLI is intrinsic for CLI inheritance to be supported.

With CLI inheritance support, you need not explicitly configure a parameter for an area or interface. In **Cisco IOS XR Software**, the parameters of interfaces in the same area can be exclusively configured with a single command, or parameter values can be inherited from a higher hierarchical level—such as from the area configuration level or the router ospf configuration levels.

For example, the hello interval value for an interface is determined by this precedence "IF" statement:

If the `hello interval` command is configured at the interface configuration level, then use the interface configured value, else

If the `hello interval` command is configured at the area configuration level, then use the area configured value, else

If the `hello interval` command is configured at the router ospf configuration level, then use the router ospf configured value, else

Use the default value of the command.

---

**Tip**

Understanding hierarchical CLI and CLI inheritance saves you considerable configuration time. See *Configuring Authentication at Different Hierarchical Levels for OSPF Version 2*, on page 322 to understand how to implement these fundamentals. In addition, **Cisco IOS XR Software** examples are provided in *Configuration Examples for Implementing OSPF*, on page 370.

### **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.
This figure illustrates the routing components in an OSPF network topology.

**Figure 15: OSPF Routing Components**

**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 294 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 294, 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 \textit{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 294, 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 294, 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 294, 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 294) is attached to one area (for example, all the interfaces reside in the same area).

OSPF Process and Router ID

An OSPF process is a logical routing entity running OSPF in a physical router. This logical routing entity should not be confused with the logical routing feature that allows a system administrator (known as the Cisco IOS XR Software Owner) to partition the physical box into separate routers.

A physical router can run multiple OSPF processes, although the only reason to do so would be to connect two or more OSPF domains. Each process has its own link-state database. The routes in the routing table are calculated from the link-state database. One OSPF process does not share routes with another OSPF process unless the routes are redistributed.

Each OSPF process is identified by a router ID. The router ID must be unique across the entire routing domain. OSPF obtains a router ID from the following sources, in order of decreasing preference:

- By default, when the OSPF process initializes, it checks if there is a router-id in the checkpointing database.
- The 32-bit numeric value specified by the OSPF router-id command in router configuration mode. (This value can be any 32-bit value. It is not restricted to the IPv4 addresses assigned to interfaces on this router, and need not be a routable IPv4 address.)
- The ITAL selected router-id.
- The primary IPv4 address of an interface over which this OSPF process is running. The first interface address in the OSPF interface is selected.

We recommend that the router ID be set by the router-id command in router configuration mode. Separate OSPF processes could share the same router ID, in which case they cannot reside in the same OSPF routing domain.
Supported OSPF Network Types

OSPF classifies different media into the following types of networks:

- NBMA networks
- Point-to-point networks (POS)
- Broadcast networks (Gigabit Ethernet)
- Point-to-multipoint

You can configure your Cisco IOS XR network as either a broadcast or an NBMA network. Using this feature, you can configure broadcast networks as NBMA networks when, for example, you have routers in your network that do not support multicast addressing.

Route Authentication Methods for OSPF

OSPF Version 2 supports two types of authentication: plain text authentication and MD5 authentication. By default, no authentication is enabled (referred to as null authentication in RFC 2178).

OSPV 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 311.

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). 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 flooded routing updates to immediate neighbors. No DR or backup DR (BDR) exists; all routing information is flooded to each router.

On broadcast or NBMA segments only, OSPF minimizes the amount of information being exchanged on a segment by choosing one router to be a DR and one router to be a BDR. Thus, the routers on the segment have a central point of contact for information exchange. Instead of each router exchanging routing updates with every other router on the segment, each router exchanges information with the DR and BDR. The DR and BDR relay the information to the other routers. On broadcast network segments the number of OSPF packets is further reduced by the DR and BDR sending such OSPF updates to a multicast IP address that all OSPF routers on the network segment are listening on.

The software looks at the priority of the routers on the segment to determine which routers are the DR and BDR. The router with the highest priority is elected the DR. If there is a tie, then the router with the higher router ID takes precedence. After the DR is elected, the BDR is elected the same way. A router with a router priority set to zero is ineligible to become the DR or BDR.

**Default Route for OSPF**

Type 5 (ASE) LSAs are generated and flooded to all areas except stub areas. For the routers in a stub area to be able to route packets to destinations outside the stub area, a default route is injected by the ABR attached to the stub area.

The cost of the default route is 1 (default) or is determined by the value specified in the `default-cost` command.
Link-State Advertisement Types for OSPF Version 2

Each of the following LSA types has a different purpose:

- **Router LSA (Type 1)**—Describes the links that the router has within a single area, and the cost of each link. These LSAs are flooded within an area only. The LSA indicates if the router can compute paths based on quality of service (QoS), whether it is an ABR or ASBR, and if it is one end of a virtual link. Type 1 LSAs are also used to advertise stub networks.

- **Network LSA (Type 2)**—Describes the link state and cost information for all routers attached to a multiaccess network segment. This LSA lists all the routers that have interfaces attached to the network segment. It is the job of the designated router of a network segment to generate and track the contents of this LSA.

- **Summary LSA for ABRs (Type 3)**—Advertises internal networks to routers in other areas (interarea routes). Type 3 LSAs may represent a single network or a set of networks aggregated into one prefix. Only ABRs generate summary LSAs.

- **Summary LSA for ASBRs (Type 4)**—Advertises an ASBR and the cost to reach it. Routers that are trying to reach an external network use these advertisements to determine the best path to the next hop. ABRs generate Type 4 LSAs.

- **Autonomous system external LSA (Type 5)**—Redistributes routes from another autonomous system, usually from a different routing protocol into OSPF.

- **Autonomous system external LSA (Type 7)**—Provides for carrying external route information within an NSSA. Type 7 LSAs may be originated by and advertised throughout an NSSA. NSSAs do not receive or originate Type 5 LSAs. Type 7 LSAs are advertised only within a single NSSA. They are not flooded into the backbone area or into any other area by border routers.

- **Intra-area-prefix LSAs (Type 9)**—A router can originate multiple intra-area-prefix LSAs for every router or transit network, each with a unique link-state ID. The link-state ID for each intra-area-prefix LSA describes its association to either the router LSA or network LSA and contains prefixes for stub and transit networks.

- **Area local scope (Type 10)**—Opaque LSAs are not flooded past the borders of their associated area.

- **Link-state (Type 11)**—The LSA is flooded throughout the AS. The flooding scope of Type 11 LSAs are equivalent to the flooding scope of AS-external (Type 5) LSAs. Similar to Type 5 LSAs, the LSA is rejected if a Type 11 opaque LSA is received in a stub area from a neighboring router within the stub area. Type 11 opaque LSAs have these attributes:
  - LSAs are flooded throughout all transit areas.
  - LSAs are not flooded into stub areas from the backbone.
  - LSAs are not originated by routers into their connected stub areas.


Link-State Advertisement Types for OSPFv3

Each of the following LSA types has a different purpose:
• Router LSA (Type 1)—Describes the link state and costs of a the router link to the area. These LSAs are flooded within an area only. The LSA indicates whether the router is an ABR or ASBR and if it is one end of a virtual link. Type 1 LSAs are also used to advertise stub networks. In OSPFv3, these LSAs have no address information and are network protocol independent. In OSPFv3, router interface information may be spread across multiple router LSAs. Receivers must concatenate all router LSAs originated by a given router before running the SPF calculation.

• Network LSA (Type 2)—Describes the link state and cost information for all routers attached to a multiaccess network segment. This LSA lists all OSPF routers that have interfaces attached to the network segment. Only the elected designated router for the network segment can generate and track the network LSA for the segment. In OSPFv3, network LSAs have no address information and are network-protocol-independent.

• Interarea-prefix LSA for ABRs (Type 3)—Advertises internal networks to routers in other areas (interarea routes). Type 3 LSAs may represent a single network or set of networks aggregated into one prefix. Only ABRs generate Type 3 LSAs. In OSPFv3, addresses for these LSAs are expressed as "prefix and prefix length" instead of "address and mask." The default route is expressed as a prefix with length 0.

• Interarea-router LSA for ASBRs (Type 4)—Advertises an ASBR and the cost to reach it. Routers that are trying to reach an external network use these advertisements to determine the best path to the next hop. ABRs generate Type 4 LSAs.

• Autonomous system external LSA (Type 5)—Redistributes routes from another autonomous system, usually from a different routing protocol into OSPF. In OSPFv3, addresses for these LSAs are expressed as "prefix and prefix length" instead of "address and mask." The default route is expressed as a prefix with length 0.

• Autonomous system external LSA (Type 7)—Provides for carrying external route information within an NSSA. Type 7 LSAs may be originated by and advertised throughout an NSSA. NSSAs do not receive or originate Type 5 LSAs. Type 7 LSAs are advertised only within a single NSSA. They are not flooded into the backbone area or into any other area by border routers.

• Link LSA (Type 8)—Has link-local flooding scope and is never flooded beyond the link with which it is associated. Link LSAs provide the link-local address of the router to all other routers attached to the link or network segment, inform other routers attached to the link of a list of IPv6 prefixes to associate with the link, and allow the router to assert a collection of Options bits to associate with the network LSA that is originated for the link.

• Intra-area-prefix LSAs (Type 9)—A router can originate multiple intra-area-prefix LSAs for every router or transit network, each with a unique link-state ID. The link-state ID for each intra-area-prefix LSA describes its association to either the router LSA or network LSA and contains prefixes for stub and transit networks.

An address prefix occurs in almost all newly defined LSAs. The prefix is represented by three fields: Prefix Length, Prefix Options, and Address Prefix. In OSPFv3, addresses for these LSAs are expressed as "prefix and prefix length" instead of "address and mask." The default route is expressed as a prefix with length 0.

Inter-area-prefix and intra-area-prefix LSAs carry all IPv6 prefix information that, in IPv4, is included in router LSAs and network LSAs. The Options field in certain LSAs (router LSAs, network LSAs, interarea-router LSAs, and link LSAs) has been expanded to 24 bits to provide support for OSPF in IPv6.

In OSPFv3, the sole function of link-state ID in interarea-prefix LSAs, interarea-router LSAs, and autonomous system external LSAs is to identify individual pieces of the link-state database. All addresses or router IDs that are expressed by the link-state ID in OSPF Version 2 are carried in the body of the LSA in OSPFv3.
Virtual Link and Transit Area for OSPF

In OSPF, routing information from all areas is first summarized to the backbone area by ABRs. The same ABRs, in turn, propagate such received information to their attached areas. Such hierarchical distribution of routing information requires that all areas be connected to the backbone area (Area 0). Occasions might exist for which an area must be defined, but it cannot be physically connected to Area 0. Examples of such an occasion might be if your company makes a new acquisition that includes an OSPF area, or if Area 0 itself is partitioned.

In the case in which an area cannot be connected to Area 0, you must configure a virtual link between that area and Area 0. The two endpoints of a virtual link are ABRs, and the virtual link must be configured in both routers. The common nonbackbone area to which the two routers belong is called a transit area. A virtual link specifies the transit area and the router ID of the other virtual endpoint (the other ABR).

A virtual link cannot be configured through a stub area or NSSA.

This figure illustrates a virtual link from Area 3 to Area 0.

Figure 16: Virtual Link to Area 0

### Passive Interface

Setting an interface as passive disables the sending of routing updates for the neighbors, hence adjacencies will not be formed in OSPF. However, the particular subnet will continue to be advertised to OSPF neighbors. Use the `passive` command in appropriate mode to suppress the sending of OSPF protocol operation on an interface.

It is recommended to use passive configuration on interfaces that are connecting LAN segments with hosts to the rest of the network, but are not meant to be transit links between routers.
OSPFv2 Sham Link Support for MPLS VPN

In an MPLS VPN environment, several VPN client sites can be connected in the same OSPF area. If these sites are connected over a backdoor link (intra-area link) and connected over the VPN backbone, all traffic passes over the backdoor link instead of over the VPN backbone, because provider edge routers advertise OSPF routes learned over the VPN backbone as inter-area or external routes that are less preferred than intra-area routes advertised over backdoor links.

To correct this default OSPF behavior in an MPLS VPN, configure a sham link between two provider edge (PE) routers to connect the sites through the MPLS VPN backbone. A sham link represents an intra-area (unnumbered point-to-point) connection between PE routers. All other routers in the area see the sham link and use it to calculate intra-area shortest path first (SPF) routes to the remote site. A cost must be configured with each sham link to determine whether traffic is sent over the backdoor link or sham link.

Configured source and destination addresses serve as the endpoints of the sham link. The source and destination IP addresses must belong to the VRF and must be advertised by Border Gateway Protocol (BGP) as host routes to remote PE routers. The sham-link endpoint addresses should not be advertised by OSPF.

For example, Figure 17: Backdoor Paths Between OSPF Client Sites, on page 302 shows three client sites, each with backdoor links. Because each site runs OSPF within Area 1 configuration, all routing between the sites follows the intra-area path across the backdoor links instead of over the MPLS VPN backbone.

If the backdoor links between the sites are used only for backup purposes, default route selection over the backbone link is not acceptable as it creates undesirable traffic flow. To establish the desired path selection over the MPLS backbone, an additional OSPF intra-area (sham link) link between the ingress and egress PE routers must be created.
A sham link is required between any two VPN sites that belong to the same OSPF area and share an OSPF backdoor link. If no backdoor link exists between sites, no sham link is required.

Figure 18: Sham Link Between PE Routers to Connected OSPF Client Sites

When a sham link is configured between the PE routers, the PE routers can populate the virtual routing and forwarding (VRF) table with the OSPF routes learned over the sham link. These OSPF routes have a larger administrative distance than BGP routes. If BGP routes are available, they are preferred over these OSPF routes with the high administrative distance.

**OSPF SPF Prefix Prioritization**

The OSPF SPF Prefix Prioritization feature enables an administrator to converge, in a faster mode, important prefixes during route installation.

When a large number of prefixes must be installed in the Routing Information Base (RIB) and the Forwarding Information Base (FIB), the update duration between the first and last prefix, during SPF, can be significant. In networks where time-sensitive traffic (for example, VoIP) may transit to the same router along with other traffic flows, it is important to prioritize RIB and FIB updates during SPF for these time-sensitive prefixes.

The OSPF SPF Prefix Prioritization feature provides the administrator with the ability to prioritize important prefixes to be installed, into the RIB during SPF calculations. Important prefixes converge faster among prefixes of the same route type per area. Before RIB and FIB installation, routes and prefixes are assigned to
various priority batch queues in the OSPF local RIB, based on specified route policy. The RIB priority batch
queues are classified as "critical," "high," "medium," and "low," in the order of decreasing priority.
When enabled, prefix alters the sequence of updating the RIB with this prefix priority:

**Critical > High > Medium > Low**

As soon as prefix priority is configured, /32 prefixes are no longer preferred by default; they are placed in the
low-priority queue, if they are not matched with higher-priority policies. Route policies must be devised to
retain /32s in the higher-priority queues (high-priority or medium-priority queues).

Priority is specified using route policy, which can be matched based on IP addresses or route tags. During
SPF, a prefix is checked against the specified route policy and is assigned to the appropriate RIB batch priority
queue.

These are examples of this scenario:

- If only high-priority route policy is specified, and no route policy is configured for a medium priority:
  - Permitted prefixes are assigned to a high-priority queue.
  - Unmatched prefixes, including /32s, are placed in a low-priority queue.

- If both high-priority and medium-priority route policies are specified, and no maps are specified for
critical priority:
  - Permitted prefixes matching high-priority route policy are assigned to a high-priority queue.
  - Permitted prefixes matching medium-priority route policy are placed in a medium-priority queue.
  - Unmatched prefixes, including /32s, are moved to a low-priority queue.

- If both critical-priority and high-priority route policies are specified, and no maps are specified for
medium priority:
  - Permitted prefixes matching critical-priority route policy are assigned to a critical-priority queue.
  - Permitted prefixes matching high-priority route policy are assigned to a high-priority queue.
  - Unmatched prefixes, including /32s, are placed in a low-priority queue.

- If only medium-priority route policy is specified and no maps are specified for high priority or critical
priority:
  - Permitted prefixes matching medium-priority route policy are assigned to a medium-priority queue.
  - Unmatched prefixes, including /32s, are placed in a low-priority queue.

Use the `[no] spf prefix-priority route-policy rpl` command to prioritize OSPF prefix installation into
the global RIB during SPF.

SPF prefix prioritization is disabled by default. In disabled mode, /32 prefixes are installed into the
global RIB, before other prefixes. If SPF prioritization is enabled, routes are matched against the
route-policy criteria and are assigned to the appropriate priority queue based on the SPF priority set.
Unmatched prefixes, including /32s, are placed in the low-priority queue.
If all /32s are desired in the high-priority queue or medium-priority queue, configure this single route map:

```plaintext
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 370 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.

```plaintext
timers spf 5 1000 90000
```

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.

*Figure 19: SPF Calculation Intervals Set by the timers spf Command*

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 20: Timer Intervals Reset After Topology Change Event, on page 306 that a topology change event was received after the start of the maximum wait time interval and that the SPF intervals have been reset.

**Figure 20: 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 failover. With NSF, peer networking devices do not experience routing flaps. During 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 failover 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 an RP failover, 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, 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 and helper mode. Restart mode occurs when the OSPFv3 process is doing a graceful restart. Helper mode refers to the neighbor routers that continue to forward traffic on established OSPFv3 routes while OSPFv3 is restarting on a neighboring router.

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 343, 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 308 section.

• If OSPFv3 attempts a restart too soon after the most recent restart, the OSPFv3 process is most likely crashing repeatedly, so the new graceful restart stops running. To control the period between allowable graceful restarts, use the graceful-restart interval command.

• When OSPFv3 starts a graceful restart with the first interface that comes up, a timer starts running to limit the duration (or lifetime) of the graceful restart. You can configure this period with the graceful-restart lifetime command. On each interface that comes up, a grace LSA (Type 11) is flooded to indicate to the neighboring routers that this router is attempting graceful restart. The neighbors enter into helper mode.

• The designated router and backup designated router check of the hello packet received from the restarting neighbor is bypassed, because it might not be valid.

Helper Mode

Helper mode is enabled by default. When a (helper) router receives a grace LSA (Type 11) from a router that is attempting a graceful restart, the following events occur:

• If helper mode has been disabled through the graceful-restart helper disable command, the router drops the LSA packet.

• If helper mode is enabled, the router enters helper mode if all of the following conditions are met:
  • The local router itself is not attempting a graceful restart.
• The local (helping) router has full adjacency with the sending neighbor.
• The value of \( lsage \) (link state age) in the received LSA is less than the requested grace period.
• The sender of the grace LSA is the same as the originator of the grace LSA.

• Upon entering helper mode, a router performs its helper function for a specific period of time. This time period is the lifetime value from the router that is in restart mode—minus the value of \( lsage \) in the received grace LSA. If the graceful restart succeeds in time, the helper’s timer is stopped before it expires. If the helper’s timer does expire, the adjacency to the restarting router is brought down, and normal OSPFv3 functionality resumes.
• The dead timer is not honored by the router that is in helper mode.
• A router in helper mode ceases to perform the helper function in any of the following cases:
  • The helper router is able to bring up a FULL adjacency with the restarting router.
  • The local timer for the helper function expires.

Graceful Restart Requirements and Restrictions

The requirements for supporting the Graceful Restart feature include:

• Cooperation of a router’s neighbors during a graceful restart. In relation to the router on which OSPFv3 is restarting, each router is called a helper.

• All neighbors of the router that does a graceful restart must be capable of doing a graceful restart.

• A graceful restart does not occur upon the first-time startup of a router.

• OSPFv3 neighbor information and database information are not check-pointed.

• An OSPFv3 process rebuilds adjacencies after it restarts.

• To ensure consistent databases after a restart, the OSPFv3 configuration must be identical to the configuration before the restart. (This requirement applies to self-originated information in the local database.) A graceful restart can fail if configurations change during the operation. In this case, data forwarding would be affected. OSPFv3 resumes operation by regenerating all its LSAs and resynchronizing its database with all its neighbors.

• Although IPv6 FIB tables remain unchanged during a graceful restart, these tables eventually mark the routes as stale through the use of a holddown timer. Enough time is allowed for the protocols to rebuild state information and converge.

• The router on which OSPFv3 is restarting must send OSPFv3 hellos within the dead interval of the process restart. Protocols must be able to retain adjacencies with neighbors before the adjacency dead timer expires. The default for the dead timer is 40 seconds. If hellos do not arrive on the adjacency before the dead timer expires, the router takes down the adjacency. The OSPFv3 Graceful Restart feature does not function properly if the dead timer is configured to be less than the time required to send hellos after the OSPFv3 process restarts.

• Simultaneous graceful restart sessions on multiple routers are not supported on a single network segment. If a router determines that multiple routers are in restart mode, it terminates any local graceful restart operation.
• This feature utilizes the available support for changing the purge time of existing OSPFv3 routes in the Routing Information Base (RIB). When graceful restart is enabled, the purge timer is set to 90 seconds by default. If graceful restart is disabled, the purge timer setting is 0.

• This feature has an associated grace LSA. This link-scope LSA is type 11.

• 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 Failover (FO) and be ready to function before the failure occurs. It reduces the downtime during switchover. By default, the router sends hello packets every 40 seconds.

With warm standby process for each OSPF process running on the Active Route Processor, the corresponding OSPF process must start on the Standby RP. There are no changes in configuration for this feature.

Warm-Standby is always enabled. This is an advantage for the systems running OSPFv3 as their IGP when they do RP failover.

Multicast-Intact Support for OSPF

The multicast-intact feature provides the ability to run multicast routing (PIM) when IGP shortcuts are configured and active on the router. Both OSPFv2 and IS-IS support the multicast-intact feature.

You can enable multicast-intact in the IGP when multicast routing protocols (PIM) are configured and IGP shortcuts are configured on the router. IGP shortcuts are MPLS tunnels that are exposed to IGP. The IGP routes IP traffic over these tunnels to destinations that are downstream from the egress router of the tunnel.
(from an SPF perspective). PIM cannot use IGP shortcuts for propagating PIM joins, because reverse path forwarding (RPF) cannot work across a unidirectional tunnel.

When you enable multicast-intact on an IGP, the IGP publishes a parallel or alternate set of equal-cost next hops for use by PIM. These next hops are called mcast-intact next hops. The mcast-intact next hops have the following attributes:

- They are guaranteed not to contain any IGP shortcuts.
- They are not used for unicast routing but are used only by PIM to look up an IPv4 next-hop to a PIM source.
- They are not published to the FIB.
- When multicast-intact is enabled on an IGP, all IPv4 destinations that were learned through link-state advertisements are published with a set equal-cost mcast-intact next hops to the RIB. This attribute applies even when the native next hops have no IGP shortcuts.

In OSPF, the max-paths (number of equal-cost next hops) limit is applied separately to the native and mcast-intact next hops. The number of equal cost mcast-intact next hops is the same as that configured for the native next hops.

Load Balancing in OSPF Version 2 and OSPFv3

When a router learns multiple routes to a specific network by using multiple routing processes (or routing protocols), it installs the route with the lowest administrative distance in the routing table. Sometimes the router must select a route from among many learned by using the same routing process with the same administrative distance. In this case, the router chooses the path with the lowest cost (or metric) to the destination. Each routing process calculates its cost differently; the costs may need to be manipulated to achieve load balancing.

OSPF performs load balancing automatically. If OSPF finds that it can reach a destination through more than one interface and each path has the same cost, it installs each path in the routing table. The only restriction on the number of paths to the same destination is controlled by the maximum-paths (OSPF) command.

The range for maximum paths is from 1 to 8 and the default number of maximum paths is 8.

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 OSPF point-to-point type using the network point-to-point command to enable the interface for a multi-area adjacency.

• Inherits the Bidirectional Forwarding Detection (BFD) characteristics from its primary interface. BFD is not configurable under a multi-area interface; however, it is configurable under the primary interface.

The multi-area interface inherits the interface characteristics from its primary interface, but some interface characteristics can be configured under the multi-area interface configuration mode as shown below:

```
RP/0/RSP0/CPU0# multi-area-interface GigabitEthernet 0/1/0/3
RP/0/RSP0/CPU0#?
```

```
authentication       Enable authentication
authentication-key   Authentication password (key)
cost                 Interface cost
cost-fallback        Cost when cumulative bandwidth goes below the threshold
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/RSP0/CPU0#?
```

### Label Distribution Protocol IGP Auto-configuration for OSPF

Label Distribution Protocol (LDP) Interior Gateway Protocol (IGP) auto-configuration simplifies the procedure to enable LDP on a set of interfaces used by an IGP instance, such as OSPF. LDP IGP auto-configuration can be used on a large number of interfaces (for example, when LDP is used for transport in the core) and on multiple OSPF instances simultaneously.

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

LDP IGP auto-configuration can also be explicitly disabled on an individual interface basis under LDP using the `igp auto-config disable` command. This allows LDP to receive all OSPF interfaces minus the ones explicitly disabled.

See Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide for information on configuring LDP IGP auto-configuration.

### OSPF Authentication Message Digest Management

All OSPF routing protocol exchanges are authenticated and the method used can vary depending on how authentication is configured. When using cryptographic authentication, the OSPF routing protocol uses the Message Digest 5 (MD5) authentication algorithm to authenticate packets transmitted between neighbors in the network. For each OSPF protocol packet, a key is used to generate and verify a message digest that is appended to the end of the OSPF packet. The message digest is a one-way function of the OSPF protocol.
packet and the secret key. Each key is identified by the combination of interface used and the key identification. An interface may have multiple keys active at any time.

To manage the rollover of keys and enhance MD5 authentication for OSPF, you can configure a container of keys called a keychain with each key comprising the following attributes: generate/accept time, key identification, and authentication algorithm.

**GTSM TTL Security Mechanism for OSPF**

OSPF is a link state protocol that requires networking devices to detect topological changes in the network, flood Link State Advertisement (LSA) updates to neighbors, and quickly converge on a new view of the topology. However, during the act of receiving LSAs from neighbors, network attacks can occur, because there are no checks that unicast or multicast packets are originating from a neighbor that is one hop away or multiple hops away over virtual links.

For virtual links, OSPF packets travel multiple hops across the network; hence, the TTL value can be decremented several times. For these type of links, a minimum TTL value must be allowed and accepted for multiple-hop packets.

To filter network attacks originating from invalid sources traveling over multiple hops, the Generalized TTL Security Mechanism (GTSM), RFC 3682, is used to prevent the attacks. GTSM filters link-local addresses and allows for only one-hop neighbor adjacencies through the configuration of TTL value 255. The TTL value in the IP header is set to 255 when OSPF packets are originated, and checked on the received OSPF packets against the default GTSM TTL value 255 or the user configured GTSM TTL value, blocking unauthorized OSPF packets originated from TTL hops away.

**Path Computation Element for OSPFv2**

A PCE is an entity (component, application, or network node) that is capable of computing a network path or route based on a network graph and applying computational constraints.

PCE is accomplished when a PCE address and client is configured for MPLS-TE. PCE communicates its PCE address and capabilities to OSPF then OSPF packages this information in the PCE Discovery type-length-value (TLV) (Type 2) and reoriginate the RI LSA. OSPF also includes the Router Capabilities TLV (Type 1) in all its RI LSAs. The PCE Discovery TLV contains the PCE address sub-TLV (Type 1) and the Path Scope Sub-TLV (Type 2).

The PCE Address Sub-TLV specifies the IP address that must be used to reach the PCE. It should be a loop-back address that is always reachable, this TLV is mandatory, and must be present within the PCE Discovery TLV. The Path Scope Sub-TLV indicates the PCE path computation scopes, which refers to the PCE ability to compute or participate in the computation of intra-area, inter-area, inter-AS or inter-layer TE LSPs.

PCE extensions to OSPFv2 include support for the Router Information Link State Advertisement (RI LSA). OSPFv2 is extended to receive all area scopes (LSA Types 9, 10, and 11). However, OSPFv2 originates only area scope Type 10.

For detailed information for the Path Computation Element feature see the Implementing MPLS Traffic Engineering on Cisco ASR 9000 Series Router module of the Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide and the following IETF drafts:

- draft-ietf-ospf-cap-09
- draft-ietf-pce-disco-proto-ospf-00
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

Management Information Base (MIB) for OSPFv3

Cisco IOS XR supports full MIBs and traps for OSPFv3, as defined in RFC 5643. The RFC 5643 defines objects of the Management Information Base (MIB) for use with the Open Shortest Path First (OSPF) Routing Protocol for IPv6 (OSPF version 3).

The OSPFv3 MIB implementation is based on the IETF draft Management Information Base for OSPFv3 (draft-ietf-ospf-ospfv3-mib-8). Users need to update the NMS application to pick up the new MIB when upgraded to RFC 5643.

Refer to the Cisco ASR 9000 Series Aggregation Services Router MIB Specification Guide for more information on Cisco IOS XR MIB support.

Multiple OSPFv3 Instances

SNMPv3 supports "contexts" that can be used to implement MIB views on multiple OSPFv3 instances, in the same system.

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 all the reachable OSPF prefixes or all the prefixes permitted by the prefix-list, if the prefix-list option is used. The UCMP computation happens only after the primary SPF and route calculation is completed. There would be a configurable delay (default delay is 100 ms) from the time primary route calculation is completed and UCMP computation is started. Use the UCMP `delay-interval` command to configure the delay between primary SPF completion and start of UCMP computation. UCMP computation will be done during the fast re-route computation (IPFRR does not need to be enabled for UCMP computation to be performed). If IPFRR is enabled, the fast re-route backup paths will be calculated for both the primary equal cost multipath (ECMP) paths and the UCMP paths.

To manually adjust UCMP ratio, use any command that changes the metric of the link.

- By using the bandwidth command in interface configuration mode
- By adjusting the OSPF interface cost on the link

How to Implement OSPF

This section contains the following procedures:

Enabling OSPF

This task explains how to perform the minimum OSPF configuration on your router that is to enable an OSPF process with a router ID, configure a backbone or nonbackbone area, and then assign one or more interfaces on which OSPF runs.

Before You Begin

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

SUMMARY STEPS

1. configure
2. Do one of the following:
   • `router ospf` `process-name`
   • `router ospfv3` `process-name`
3. `router-id` `{ router-id }
4. `area` `area-id`
5. `interface` `type interface-path-id`
6. Repeat Step 5 for each interface that uses OSPF.
7. `log adjacency changes` `[ detail ] [ enable | disable ]`
8. `commit`
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
</tbody>
</table>
| Step 2 | Do one of the following:  
  • `router ospf process-name`  
  • `router ospfv3 process-name` | 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. |
| Example:  
  RP/0/RSP0/CPU0:router(config)# router ospf 1  
  or  
  RP/0/RSP0/CPU0:router(config)# router ospfv3 1 | |
| Step 3 | `router-id { router-id }` | Configures a router ID for the OSPF process.  
  Note  
  We recommend using a stable IP address as the router ID. |
| Example:  
  RP/0/RSP0/CPU0:router(config-ospf)# router-id 192.168.4.3 | |
| Step 4 | `area area-id` | 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. |
| Example:  
  RP/0/RSP0/CPU0:router(config-ospf)# area 0 | |
| Step 5 | `interface type interface-path-id` | Enters interface configuration mode and associates one or more interfaces for the area configured in Step 4. |
| Example:  
  RP/0/RSP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/1/0/3 | |
| Step 6 | Repeat Step 5 for each interface that uses OSPF. | |
| Step 7 | `log adjacency changes [ detail ] [ enable | disable ]` | (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 |
Purpose

Command or Action | Purpose
--- | ---
Step 8 | commit

---

Configuring Stub and Not-So-Stubby Area Types

This task explains how to configure the stub area and the NSSA for OSPF.

**SUMMARY STEPS**

1. `configure`
2. Do one of the following:
   - `router ospf  process-name`
   - `router ospfv3  process-name`
3. `router-id  {  router-id  }
4. `area  area-id`
5. Do one of the following:
   - `stub  [  no-summary  ]`
   - `nssa  [  no-redistribution  ]  [  default-information-originate  ]  [  no-summary  ]`
6. Do one of the following:
   - `stub`
   - `nssa`
7. `default-cost  cost`
8. `commit`
9. Repeat this task on all other routers in the stub area or NSSA.

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>configure</code></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Do one of the following:</td>
</tr>
<tr>
<td></td>
<td>* <code>router ospf  process-name</code></td>
</tr>
<tr>
<td></td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
</tbody>
</table>

---

Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x

OL-26648-03
### Implementing OSPF

#### Configuring Stub and Not-So-Stubby Area Types

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <code>router ospfv3 process-name</code></td>
<td>Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Note</strong> The <code>process-name</code> argument is any alphanumeric string no longer than 40 characters.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router ospf 1 or</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router ospfv3 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>router-id { router-id }</code></td>
<td>Configures a router ID for the OSPF process.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Note</strong> We recommend using a stable IP address as the router ID.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf)# router-id 192.168.4.3</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> <code>area area-id</code></td>
<td>Enters area configuration mode and configures a nonbackbone area for the OSPF process.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Note</strong> The <code>area-id</code> 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.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf)# area 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> Do one of the following:</td>
<td>Defines the nonbackbone area as a stub area.</td>
</tr>
<tr>
<td>• stub [ no-summary ]</td>
<td><strong>Note</strong> Specify the <code>no-summary</code> 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.</td>
</tr>
<tr>
<td>• nssa [ no-redistribution ] [ default-information-originate ] [ no-summary ]</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# stub no summary or</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# nssa no-redistribution</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> Do one of the following:</td>
<td>(Optional) Turns off the options configured for stub and NSSA areas.</td>
</tr>
<tr>
<td>• stub</td>
<td><strong>Note</strong> If you configured the stub and NSSA areas using the optional keywords ( <code>no-summary</code>, <code>no-redistribution</code>, <code>default-information-originate</code>, and <code>no-summary</code> ) in Step 5, you must now reissue the <code>stub</code> and <code>nssa</code> commands without the keywords—rather than using the <code>no</code> form of the command.</td>
</tr>
<tr>
<td>• nssa</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# stub</td>
<td></td>
</tr>
</tbody>
</table>
### 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. interface  type interface-path-id
13. Do one of the following:
   • neighbor  ip-address  [ priority number ] [ poll-interval  seconds ] [ cost number ] [ database-filter  [ all ] ]
   • neighbor  ipv6-link-local-address  [ priority number ] [ poll-interval  seconds ] [ cost number ] [ database-filter  [ all ] ]
14. Repeat Step 13 for all neighbors on the interface.
15. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
</tbody>
</table>
| Step 2 Do one of the following:  
  • router ospf  process-name  
  • router ospfv3  process-name | Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode. |
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router ospf 1 or RP/0/RSP0/CPU0:router(config)# router ospfv3 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td>router-id { router-id }</td>
<td>Configures a router ID for the OSPF process.</td>
</tr>
<tr>
<td>Example:</td>
<td>Note: We recommend using a stable IP address as the router ID.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf)# router-id 192.168.4.3</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>area area-id</td>
<td>Enters area configuration mode and configures an area for the OSPF process.</td>
</tr>
<tr>
<td>Example:</td>
<td>• The example configures a backbone area.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf)# area 0</td>
<td>• 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.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td>network { broadcast</td>
<td>non-broadcast</td>
</tr>
<tr>
<td>Example:</td>
<td>• The example sets the network type to NBMA.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# network non-broadcast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td>dead-interval seconds</td>
<td>(Optional) Sets the time to wait for a hello packet from a neighbor before declaring the neighbor down.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# dead-interval 40</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td></td>
</tr>
<tr>
<td>hello-interval seconds</td>
<td>(Optional) Specifies the interval between hello packets that OSPF sends on the interface.</td>
</tr>
<tr>
<td>Example:</td>
<td>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.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# hello-interval 10</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td></td>
</tr>
<tr>
<td>interface type interface-path-id</td>
<td>Enters interface configuration mode and associates one or more interfaces for the area configured in Step 4.</td>
</tr>
<tr>
<td>Example:</td>
<td>• 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.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/2/0/0</td>
<td></td>
</tr>
</tbody>
</table>
### 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/RSP0/CPU0:router(config-ospf-ar-if)# neighbor 10.20.20.1 priority 3 poll-interval 15

or

RP/0/RSP0/CPU0:router(config-ospf-ar-if)# neighbor fe80::3203:a0ff:fe9d:f3fe

**Purpose**

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 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.

**Purpose**

- 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.
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 13</strong></td>
<td>Do one of the following:</td>
</tr>
<tr>
<td>• neighbor ip-address [priority number] [poll-interval seconds] [cost number] [database-filter [all]]</td>
<td>Configures the IPv4 address of OSPF neighbors interconnecting to nonbroadcast networks.</td>
</tr>
<tr>
<td>• neighbor ipv6-link-local-address [priority number] [poll-interval seconds] [cost number] [database-filter [all]]</td>
<td>Configures the link-local IPv6 address of OSPFv3 neighbors.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0 /CPU0:router(config-ospf-ar)# neighbor 10.34.16.6</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0 /CPU0:router(config-ospf-ar)# neighbor fe80::3203:a0ff:fe9d:f3f</td>
<td></td>
</tr>
</tbody>
</table>

- 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 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” or routing loops.

<table>
<thead>
<tr>
<th>Step 14</th>
<th>Repeat Step 13 for all neighbors on the interface.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 15</td>
<td>commit</td>
</tr>
</tbody>
</table>

**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.
Authentication configured at the interface level overrides authentication configured at the area level and the router process level. If an interface does not have authentication specifically configured, the interface inherits the authentication parameter value from a higher hierarchical level. See OSPF Hierarchical CLI and CLI Inheritance, on page 293 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 297 for information about each type of authentication and when you should use a specific method for your network.

**SUMMARY STEPS**

1. configure
2. router ospf  process-name
3. router-id  { router-id }
4. authentication  [ message-digest | null ]
5. message-digest-key  key-id  md5  { key | clear key | encrypted key | LINE}
6. area  area-id
7. interface  type interface-path-id
8. Repeat Step 7 for each interface that must communicate, using the same authentication.
9. exit
10. area  area-id
11. authentication  [ message-digest | null ]
12. interface  type interface-path-id
13. Repeat Step 12 for each interface that must communicate, using the same authentication.
14. interface  type interface-path-id
15. authentication  [ message-digest | null ]
16. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router ospf  process-name</td>
<td></td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# router ospf 1</td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong> The  process-name argument is any alphanumeric string no longer than 40 characters.</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>router-id</strong> { router-id }</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf)# router-id 192.168.4.3</td>
</tr>
<tr>
<td>Purpose</td>
<td>Configures a router ID for the OSPF process.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>authentication</strong> [ message-digest</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf)#authentication message-digest</td>
</tr>
<tr>
<td>Purpose</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>message-digest-key</strong> { key-id } md5 { key</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf)#message-digest-key 4 md5 yourkey</td>
</tr>
<tr>
<td>Purpose</td>
<td>Specifies the MD5 authentication key for the OSPF process. • The neighbor routers must have the same key identifier.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>area</strong> area-id</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf)# area 0</td>
</tr>
<tr>
<td>Purpose</td>
<td>Enters area configuration mode and configures a backbone area for the OSPF process.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>interface</strong> type interface-path-id</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/1/0/3</td>
</tr>
<tr>
<td>Purpose</td>
<td>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).</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>Repeat Step 7 for each interface that must communicate, using the same authentication.</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td><strong>exit</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# exit</td>
</tr>
<tr>
<td>Purpose</td>
<td>Enters area OSPF configuration mode.</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td><strong>area</strong> area-id</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf)# area 1</td>
</tr>
<tr>
<td>Purpose</td>
<td>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.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>**authentication [ message-digest</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# authentication</td>
</tr>
<tr>
<td></td>
<td>Enables Type 1 (plain text) authentication that provides no security.</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td><strong>Step 12</strong></td>
<td><strong>interface type interface-path-id</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/1/0/0</td>
</tr>
<tr>
<td></td>
<td>Enters interface configuration mode and associates one or more interfaces to the nonbackbone area 1 specified in Step 7.</td>
</tr>
<tr>
<td></td>
<td>• All interfaces configured inherit the authentication parameter values configured for area 1.</td>
</tr>
<tr>
<td><strong>Step 13</strong></td>
<td>Repeat Step 12 for each interface that must communicate, using the same authentication.</td>
</tr>
<tr>
<td><strong>Step 14</strong></td>
<td><strong>interface type interface-path-id</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/3/0/0</td>
</tr>
<tr>
<td></td>
<td>Enters interface configuration mode and associates one or more interfaces to a different authentication type.</td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td>**authentication [ message-digest</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar-if)# authentication null</td>
</tr>
<tr>
<td></td>
<td>Specifies no authentication on GigabitEthernet interface 0/3/0/0, overriding the plain text authentication specified for area 1.</td>
</tr>
<tr>
<td></td>
<td>• By default, all of the interfaces configured in the same area inherit the same authentication parameter values of the area.</td>
</tr>
<tr>
<td><strong>Step 16</strong></td>
<td><strong>commit</strong></td>
</tr>
</tbody>
</table>

**Controlling the Frequency That the Same LSA Is Originated or Accepted for OSPF**

This task explains how to tune the convergence time of OSPF routes in the routing table when many LSAs need to be flooded in a very short time interval.
SUMMARY STEPS

1. configure
2. Do one of the following:
   • `router ospf process-name`
   • `router ospfv3 process-name`
3. `router-id { router-id }
4. Perform Step 5 or Step 6 or both to control the frequency that the same LSA is originated or accepted.
5. `timers lsa refresh seconds`
6. `timers lsa min-arrival seconds`
7. `timers lsa group-pacing seconds`
8. `commit`

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> Do one of the following:</td>
<td>Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.</td>
</tr>
<tr>
<td>• <code>router ospf process-name</code></td>
<td>or</td>
</tr>
<tr>
<td>• <code>router ospfv3 process-name</code></td>
<td>Note The <code>process-name</code> argument is any alphanumeri</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Step 6</strong> timers lsa min-arrival seconds</td>
<td>Limits the frequency that new processes of any particular OSPF Version 2 LSA can be accepted during flooding.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf)# timers lsa min-arrival 2</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> timers lsa group-pacing seconds</td>
<td>Changes the interval at which OSPF link-state LSAs are collected into a group for flooding.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf)# timers lsa group-pacing 1000</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>

### 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 301.

#### 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 375.

#### 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 296 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.
If you decide to configure plain text authentication or no authentication, see the `authentication` command provided in *OSPF Commands on Cisco ASR 9000 Series Router module in Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference.*

### SUMMARY STEPS

1. Do one of the following:
   - `show ospf [process-name]`
   - `show ospfv3 [process-name]`
2. `configure`
3. Do one of the following:
   - `router ospf process-name`
   - `router ospfv3 process-name`
4. `router-id {router-id}`
5. `area area-id`
6. `virtual-link router-id`
7. `authentication message-digest`
8. `message-digest-key key-id md5 {key | clear key | encrypted key}`
9. Repeat all of the steps in this task on the ABR that is at the other end of the virtual link. Specify the same key ID and key that you specified for the virtual link on this router.
10. `commit`
11. Do one of the following:
    - `show ospf [process-name] [area-id] virtual-links`
    - `show ospfv3 [process-name] virtual-links`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Do one of the following:</td>
</tr>
<tr>
<td>• <code>show ospf [process-name]</code></td>
<td>(Optional) Displays general information about OSPF routing processes.</td>
</tr>
<tr>
<td>• <code>show ospfv3 [process-name]</code></td>
<td>• The output displays the router ID of the local router. You need this router ID to configure the other end of the link.</td>
</tr>
</tbody>
</table>

Example:

    RP/0/RSP0/CPU0:router# show ospf
### Implementing OSPF

#### Creating a Virtual Link with MD5 Authentication to Area 0 for OSPF

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td><strong>RP/0/RSP0/CPU0:router</strong> show ospfv3</td>
<td></td>
</tr>
</tbody>
</table>

**Step 2** configure

**Step 3** Do one of the following:

- `router ospf process-name`
- `router ospfv3 process-name`

**Example:**

- `RP/0/RSP0/CPU0:router(config)# router ospf 1`  
- `RP/0/RSP0/CPU0:router(config)# router ospfv3 1`

Purpose: 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 4** `router-id { router-id }`

**Example:**

- `RP/0/RSP0/CPU0:router(config-ospf)# router-id 192.168.4.3`

Purpose: 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/RSP0/CPU0:router(config-ospf)# area 1`

Purpose: 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:**

- `RP/0/RSP0/CPU0:router(config-ospf-ar)# virtual-link 10.3.4.5`

Purpose: Defines an OSPF virtual link.

- See .

**Step 7** `authentication message-digest`

**Example:**

- `RP/0/RSP0/CPU0:router(config-ospf-ar-vl)#authentication message-digest`

Purpose: Selects MD5 authentication for this virtual link.

**Step 8** `message-digest-key key-id md5 { key | clear key | encrypted key }

**Example:**

- `RP/0/RSP0/CPU0:router(config-ospf-ar-vl)#message-digest-key 4 md5 yourkey`

Purpose: 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.
Creating a Virtual Link with MD5 Authentication to Area 0 for OSPF

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Step 9</td>
<td>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.</td>
</tr>
<tr>
<td>Step 10</td>
<td>commit</td>
</tr>
<tr>
<td>Step 11</td>
<td>Do one of the following:</td>
</tr>
<tr>
<td></td>
<td>• show ospf [process-name] [area-id] virtual-links</td>
</tr>
<tr>
<td></td>
<td>• show ospfv3 [process-name] virtual-links</td>
</tr>
<tr>
<td>Example:</td>
<td>(Optional) Displays the parameters and the current state of OSPF virtual links.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router# show ospf 1 2 virtual-links</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router# show ospfv3 1 virtual-links</td>
<td></td>
</tr>
</tbody>
</table>

Examples

In the following example, the show ospfv3 virtual-links EXEC configuration command verifies that the OSPF_VL0 virtual link to the OSPFv3 neighbor is up, the ID of the virtual link interface is 2, and the IPv6 address of the virtual link endpoint is 2003:3000::1.

```
show ospfv3 virtual-links

Virtual Links for OSPFv3 1

Virtual Link OSPF_VL0 to router 10.0.0.3 is up
Interface ID 2, IPv6 address 2003:3000::1
Run as demand circuit
DoNotAge LSA allowed.
Transit area 0.1.20.255, via interface GigabitEthernet 0/1/0/1, Cost of using 2
Transmit Delay is 5 sec, State POINT TO POINT,
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:02
Adjacency State FULL (Hello suppressed)
Index 0/2/3, retransmission queue length 0, number of retransmission 1
First 0(0)/0(0)/0(0) Next 0(0)/0(0)/0(0)
Last retransmission scan length is 1, maximum is 1
Last retransmission scan time is 0 msec, maximum is 0 msec
```

---
Summarizing Subnetwork LSAs on an OSPF ABR

If you configured two or more subnetworks when you assigned your IP addresses to your interfaces, you might want the software to summarize (aggregate) into a single LSA all of the subnetworks that the local area advertises to another area. Such summarization would reduce the number of LSAs and thereby conserve network resources. This summarization is known as interarea route summarization. It applies to routes from within the autonomous system. It does not apply to external routes injected into OSPF by way of redistribution.

This task configures OSPF to summarize subnetworks into one LSA, by specifying that all subnetworks that fall into a range are advertised together. This task is performed on an ABR only.

SUMMARY STEPS

1. configure
2. Do one of the following:
   • router ospf  process-name
   • router ospfv3  process-name
3. router-id  { router-id }
4. area  area-id
5. Do one of the following:
   • range  ip-address mask  [ advertise | not-advertise ]
   • range  ipv6-prefix / prefix-length  [ advertise | not-advertise ]
6. interface  type interface-path-id
7. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
</tbody>
</table>
| Step 2 Do one of the following:  
  • router ospf  process-name
  • router ospfv3  process-name | Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode. |

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
### Implementing OSPF

#### Summarizing Subnetwork LSAs on an OSPF ABR

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 3</strong> <code>router-id { router-id }</code></td>
<td>Configures a router ID for the OSPF process.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-ospf)# router-id 192.168.4.3</code></td>
<td>Note: We recommend using a stable IPv4 address as the router ID.</td>
</tr>
</tbody>
</table>

- **Step 4** `area area-id`
  - **Example:** `RP/0/RSP0/CPU0:router(config-ospf)# area 10`
  - 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/RSP0/CPU0:router(config-ospf-ar)# range 192.168.0.0 255.255.0.0 advertise`
    - `RP/0/RSP0/CPU0:router(config-ospf-ar)# range 4004:f000::/32 advertise`
  - 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.
  - 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/RSP0/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** `commit`
Redistribute Routes into OSPF

This task redistributes routes from an IGP (could be a different OSPF process) into OSPF.

Before You Begin

For information about configuring routing policy, see Implementing Routing Policy on Cisco ASR 9000 Series Router module in the Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide.

SUMMARY STEPS

1. configure
2. Do one of the following:
   - router ospf  process-name
   - router ospfv3  process-name
3. router-id  { router-id }
4. redistribute  protocol  [ process-id ] [ level-1 ] [ level-1-2 ] [ level-2 ] [ metric  metric-value ] [ metric-type  type-value ] [ match  { external [ 1 | 2 ] } [ tag  tag-value ] [ route-policy  policy-name ]
5. Do one of the following:
   - summary-prefix  address  mask  [ not-advertise ] [ tag  tag ]
   - summary-prefix  ipv6-prefix  /prefix-length  [ not-advertise ] [ tag  tag ]
6. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
</tbody>
</table>
| Step 2 | Do one of the following:  
   - router ospf  process-name  
   - router ospfv3  process-name | Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode. |
|     | Example:  
   RP/0/RSP0/CP0U:router(config)# router ospf 1  
   or  
   RP/0/RSP0/CP0U:router(config)# router ospfv3 1 | Note The process-name argument is any alphanumeric string no longer than 40 characters. |
| Step 3 | router-id  { router-id } | Configures a router ID for the OSPF process. |
Redistribute Routes into OSPF

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: RRP/0/RSP0/CPU0:router(config-ospf)# router-id 192.168.4.3</td>
<td>Note We recommend using a stable IPv4 address as the router ID.</td>
</tr>
</tbody>
</table>

**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/RSP0/CPU0:router(config-ospf)# redistribute bgp 100
or
RP/0/RSP0/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 link type associated with the default route advertised into the OSPFv3 routing domain is the Type 1 external route.

**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]`

(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.

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# summary-prefix 10.1.0.0 255.255.0.0
or
RP/0/RSP0/CPU0:router(config-router)# summary-prefix 10.1.0.0 255.255.0.0
```

Note RPL is not supported for OSPFv3.
### Configuring OSPF Shortest Path First Throttling

This task explains how to configure SPF scheduling in millisecond intervals and potentially delay SPF calculations during times of network instability. This task is optional.

#### SUMMARY STEPS

1. configure
2. Do one of the following:
   - `router ospf`  `process-name`
   - `router ospfv3`  `process-name`
3. `router-id`  `{ router-id }
4. `timers throttle spf`  `spf-start spf-hold spf-max-wait`
5. `area`  `area-id`
6. `interface`  `type interface-path-id`
7. `commit`
8. Do one of the following:
   - `show ospf`  `{ process-name }
   - `show ospfv3`  `{ process-name }

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>Do one of the following:</td>
<td><strong>Purpose</strong></td>
<td></td>
</tr>
<tr>
<td>• <code>router ospf</code> <code>process-name</code></td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>or • <code>router ospfv3</code> <code>process-name</code></td>
<td>Note The <code>process-name</code> argument is any alphanumeric string no longer than 40 characters.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router ospf 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router ospfv3 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Configures a router ID for the OSPF process.</td>
<td></td>
</tr>
<tr>
<td><code>router-id { router-id }</code></td>
<td>Note We recommend using a stable IPv4 address as the router ID.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf)# router-id 192.168.4.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Sets SPF throttling timers.</td>
<td></td>
</tr>
<tr>
<td><code>timers throttle spf</code> <code>spf-start spf-hold spf-max-wait</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf)# timers throttle spf 10 4800 90000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Enters area configuration mode and configures a backbone area.</td>
<td></td>
</tr>
<tr>
<td><code>area</code> <code>area-id</code></td>
<td>• The <code>area-id</code> 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.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf)# area 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Enters interface configuration mode and associates one or more interfaces to the area.</td>
<td></td>
</tr>
<tr>
<td><code>interface</code> <code>type interface-path-id</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/1/0/3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>(Optional) Displays SPF throttling timers.</td>
<td></td>
</tr>
<tr>
<td><code>commit</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do one of the following:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• <code>show ospf</code> <code>process-name</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>or • <code>show ospfv3</code> <code>process-name</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router# show ospf 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Examples

In the following example, the `show ospf` EXEC configuration command is used to verify that the initial SPF schedule delay time, minimum hold time, and maximum wait time are configured correctly. Additional details are displayed about the OSPF process, such as the router type and redistribution of routes.

```
show ospf 1
Routing Process "ospf 1" with ID 192.168.4.3
  Supports only single TOS(TOS0) routes
  Supports opaque LSA
  It is an autonomous system boundary router
  Redistributing External Routes from,
    ospf 2
  Initial SPF schedule delay 5 msecs
  Minimum hold time between two consecutive SPF 100 msecs
  Maximum wait time between two consecutive SPF 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 DCBlitless 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
```

For a description of each output display field, see the `show ospf` command in the `OSPF Commands on Cisco ASR 9000 Series Router` module in `Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference`.

---

**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.
The following are restrictions when configuring nonstop forwarding:

- OSPF Cisco NSF for virtual links is not supported.
- Neighbors must be NSF aware.

**SUMMARY STEPS**

1. `configure`
2. `router ospf` *process-name*
3. `router-id` `{ router-id }`
4. Do one of the following:
   - `nsf cisco`
   - `nsf cisco enforce global`
5. `nsf interval` *seconds*
6. `nsfflush-delay-time` *seconds*
7. `nsflifetime` *seconds*
8. `nsfietf`
9. `commit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><code>configure</code></td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td>2.</td>
<td><code>router ospf</code> <em>process-name</em></td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode. <strong>Note</strong> The <em>process-name</em> argument is any alphanumeric string no longer than 40 characters.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config)# router ospf 1</code></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td><code>router-id</code> <code>{ router-id }</code></td>
<td>Configures a router ID for the OSPF process. <strong>Note</strong> We recommend using a stable IPv4 address as the router ID.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-ospf)# router-id 192.168.4.3</code></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Do one of the following:</td>
<td>Enables Cisco NSF operations for the OSPF process.</td>
</tr>
<tr>
<td></td>
<td>• <code>nsf cisco</code></td>
<td>Use the <code>nsf cisco</code> command without the optional <code>enforce</code> and <code>global</code> keywords to abort the NSF restart mechanism on the interfaces of detected non-NSF neighbors and allow NSF neighbors to function properly.</td>
</tr>
<tr>
<td></td>
<td>• <code>nsf cisco enforce global</code></td>
<td></td>
</tr>
</tbody>
</table>
**Configuring OSPF Version 2 for MPLS Traffic Engineering**

This task explains how to configure OSPF for MPLS TE. This task is optional.

For a description of the MPLS TE tasks and commands that allow you to configure the router to support tunnels, configure an MPLS tunnel that OSPF can use, and troubleshoot MPLS TE, see *Implementing MPLS Traffic Engineering on Cisco ASR 9000 Series Router* module of the *Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide*

**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)
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\n5. area *area-id*
6. mpls traffic-eng
7. interface type interface-path-id
8. commit
9. show ospf \[ *process-name* \] \[ *area-id* \] mpls traffic-eng \{ link | fragment \}

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router ospf <em>process-name</em></td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config)# router ospf 1</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Note</strong> The <em>process-name</em> argument is any alphanumeric string no longer than 40 characters.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> router-id { router-id }</td>
<td>Configures a router ID for the OSPF process.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-ospf)# router-id 192.168.4.3</td>
<td>Configures a router ID for the OSPF process.</td>
</tr>
<tr>
<td><strong>Note</strong> We recommend using a stable IPv4 address as the router ID.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> mpls traffic-eng router-id \interface-type interface-instance\</td>
<td>(Optional) Specifies that the traffic engineering router identifier for the node is the IP address associated with a given interface.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-ospf)# mpls traffic-eng router-id loopback 0</td>
<td>(Optional) Specifies that the traffic engineering router identifier for the node is the IP address associated with a given interface.</td>
</tr>
<tr>
<td>• This IP address is flooded to all nodes in TE LSAs.</td>
<td></td>
</tr>
<tr>
<td>• For all traffic engineering tunnels originating at other nodes and ending at this node, you must set the tunnel destination to the traffic engineering router identifier of the destination node because that is the address that the traffic engineering topology database at the tunnel head uses for its path calculation.</td>
<td></td>
</tr>
<tr>
<td>• We recommend that loopback interfaces be used for MPLS TE router ID because they are more stable than physical interfaces.</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

**Step 5**
- **area area-id**

  **Example:**
  
  RP/0/RSP0/CPU0:router(config-ospf)# area 0

  **Purpose:**
  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/RSP0/CPU0:router(config-ospf)# mpls traffic-eng

  **Purpose:**
  Configures the MPLS TE under the OSPF area.

**Step 7**
- **interface type interface-path-id**

  **Example:**
  
  RP/0/RSP0/CPU0:router(config-ospf-ar)#
  interface interface loopback0

  **Purpose:**
  Enters interface configuration mode and associates one or more interfaces to the area.

**Step 8**
- **commit**

**Step 9**
- **show ospf [process-name] [area-id] mpls traffic-eng {link | fragment}**

  **Example:**
  
  RP/0/RSP0/CPU0:router# show ospf 1 0 mpls traffic-eng link

  **Purpose:**
  (Optional) Displays information about the links and fragments available on the local router for MPLS TE.

### Examples

This section provides the following output examples:

**Sample Output for the show ospf Command Before Configuring MPLS TE**

In the following example, the **show route ospf** EXEC configuration command verifies that GigabitEthernet interface 0/3/0/0 exists and MPLS TE is not configured:

```
show route ospf 1

  O  11.0.0.0/24 [110/15] via 0.0.0.0, 3d19h, tunnel-te1
  O  192.168.0.12/32 [110/11] via 11.1.0.2, 3d19h, GigabitEthernet0/3/0/0
  O  192.168.0.13/32 [110/6] via 0.0.0.0, 3d19h, tunnel-te1
```
Sample Output for the `show ospf mpls traffic-eng` Command

In the following example, the `show ospf mpls traffic-eng` EXEC configuration command verifies that the MPLS TE fragments are configured correctly:

```
show ospf 1 mpls traffic-eng fragment
```

```
OSPF Router with ID (192.168.4.3) (Process ID 1)
Area 0 has 1 MPLS TE fragment. Area instance is 3.
MPLS router address is 192.168.4.2
Next fragment ID is 1
Fragment 0 has 1 link. Fragment instance is 3.
Fragment has 0 link the same as last update.
Fragment advertise MPLS router address
Link is associated with fragment 0. Link instance is 3
Link connected to Point-to-Point network
Link ID : 55.55.55.55
Interface Address : 192.168.50.21
Neighbor Address : 192.168.4.1
Admin Metric : 0
Maximum bandwidth : 19440000
Maximum global pool reservable bandwidth : 25000000
Maximum sub pool reservable bandwidth : 3125000
Number of Priority : 8
Global pool unreserved BW
Priority 0 : 25000000 Priority 1 : 25000000
Priority 2 : 25000000 Priority 3 : 25000000
Priority 4 : 25000000 Priority 5 : 25000000
Priority 6 : 25000000 Priority 7 : 25000000
Sub pool unreserved BW
Priority 0 : 3125000 Priority 1 : 3125000
Priority 2 : 3125000 Priority 3 : 3125000
Priority 4 : 3125000 Priority 5 : 3125000
Priority 6 : 3125000 Priority 7 : 3125000
Affinity Bit : 0
```

In the following example, the `show ospf mpls traffic-eng` EXEC configuration command verifies that the MPLS TE links on area instance 3 are configured correctly:

```
show ospf mpls traffic-eng link
```

```
OSPF Router with ID (192.168.4.1) (Process ID 1)
Area 0 has 1 MPLS TE links. Area instance is 3.
Links in hash bucket 53.
Link is associated with fragment 0. Link instance is 3
Link connected to Point-to-Point network
Link ID : 192.168.50.20
Interface Address : 192.168.20.50
Neighbor Address : 192.168.4.1
Admin Metric : 0
Maximum bandwidth : 19440000
Maximum global pool reservable bandwidth : 25000000
Maximum sub pool reservable bandwidth : 3125000
Number of Priority : 8
Global pool unreserved BW
Priority 0 : 25000000 Priority 1 : 25000000
Priority 2 : 25000000 Priority 3 : 25000000
Priority 4 : 25000000 Priority 5 : 25000000
Priority 6 : 25000000 Priority 7 : 25000000
Sub pool unreserved BW
Priority 0 : 3125000 Priority 1 : 3125000
Priority 2 : 3125000 Priority 3 : 3125000
Priority 4 : 3125000 Priority 5 : 3125000
Priority 6 : 3125000 Priority 7 : 3125000
Affinity Bit : 0
```
Sample Output for the show ospf Command After Configuring MPLS TE

In the following example, the `show route ospf` EXEC configuration command verifies that the MPLS TE tunnels replaced GigabitEthernet interface 0/3/0/0 and that configuration was performed correctly:

```plaintext
show route ospf 1
  O E2 192.168.10.0/24 [110/20] via 0.0.0.0, 00:00:15, tunnel2
  O E2 192.168.11.0/24 [110/20] via 0.0.0.0, 00:00:15, tunnel2
  O E2 192.168.1244.0/24 [110/20] via 0.0.0.0, 00:00:15, tunnel2
  O 192.168.12.0/24 [110/2] via 0.0.0.0, 00:00:15, tunnel2
```

Configuring OSPFv3 Graceful Restart

This task explains how to configure a graceful restart for an OSPFv3 process. This task is optional.

**SUMMARY STEPS**

1. `configure`
2. `router ospfv3 process-name`
3. `graceful-restart`
4. `graceful-restart lifetime`
5. `graceful-restart interval seconds`
6. `graceful-restart helper disable`
7. `commit`
8. `show ospfv3 [process-name [area-id]] database grace`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
</tbody>
</table>

Enter 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.

<table>
<thead>
<tr>
<th><strong>Step 2</strong></th>
<th><code>router ospfv3 process-name</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router ospfv3 test</td>
</tr>
</tbody>
</table>

Enables graceful restart on the current router.

<table>
<thead>
<tr>
<th><strong>Step 3</strong></th>
<th><code>graceful-restart</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospfv3)#graceful-restart</td>
</tr>
</tbody>
</table>

Specifies a maximum duration for a graceful restart.

- The default lifetime is 95 seconds.
- The range is 90 to 3600 seconds.
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 5 \(graceful\text{-}restart\ interval \, seconds\) | 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. |
| Example:                               | \(RP/0/RSP0/CPU0:router\text{-}config-ospfv3\)# graceful-restart interval 120 |
| Step 6 \(graceful\text{-}restart\ helper\ disable\) | Disables the helper capability. |
| Example:                               | \(RP/0/RSP0/CPU0:router\text{-}config-ospfv3\)# graceful-restart helper disable |
| Step 7 commit                          |                                                                         |
| Step 8 show ospfv3 [ process-name [ area-id ]] database grace | Displays the state of the graceful restart link. |
| Example:                               | \(RP/0/RSP0/CPU0:router\)# show ospfv3 1 database grace |

### 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/RSP0/CPU0:router# show ospfv3 1 database grace
Routing Process "ospfv3 1" with ID 2.2.2.2
Initial SPF schedule delay 5000 msecs
Minimum hold time between two consecutive SPFss 10000 msecs
Maximum wait time between two consecutive SPFss 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:

```plaintext
RP/0/RSP0/CPU0:router# show ospfv3 1 database grace
OSPFv3 Router with ID (2.2.2.2) (Process ID 1)

Router Link States (Area 0)
ADV Router Age Seq# Fragment ID Link count Bits
1.1.1.1 1949 0x8000000e 0 1 None
2.2.2.2 2007 0x80000011 0 1 None

Link (Type-8) Link States (Area 0)
ADV Router Age Seq# Link ID Interface
1.1.1.1 180 0x80000006 1 P00/2/0/0
2.2.2.2 2007 0x80000006 1 P00/2/0/0

Intra Area Prefix Link States (Area 0)
ADV Router Age Seq# Link ID Ref-lstype Ref-LSID
1.1.1.1 180 0x80000006 0 0x2001 0
2.2.2.2 2007 0x80000006 0 0x2001 0

Grace (Type-11) Link States (Area 0)
ADV Router Age Seq# Link ID Interface
2.2.2.2 2007 0x80000005 1 P00/2/0/0
```

Configuring an OSPFv2 Sham Link

This task explains how to configure a provider edge (PE) router to establish an OSPFv2 sham link connection across a VPN backbone. This task is optional.

Before You Begin

Before configuring a sham link in a Multiprotocol Label Switching (MPLS) VPN between provider edge (PE) routers, OSPF must be enabled as follows:

- Create an OSPF routing process.
- Configure a loopback interface that belongs to VRF and assign a IPv4 address with the host mask to it.
- Configure the sham link under the area submode.

See Enabling OSPF, on page 314 for information on these OSPF configuration prerequisites.
SUMMARY STEPS

1. configure
2. interface type interface-path-id
3. vrf vrf-name
4. ipv4 address ip-address mask
5. end
6. router ospf instance-id
7. vrf vrf-name
8. router-id { router-id }
9. redistribute bgp process-id
10. area area-id
11. sham-link source-address destination-address
12. cost cost
13. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 interface</td>
<td>Enters interface configuration mode.</td>
</tr>
<tr>
<td>type interface-path-id</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# interface loopback 3</td>
</tr>
<tr>
<td>Step 3 vrf</td>
<td>Assigns an interface to the VPN routing and forwarding (VRF) instance.</td>
</tr>
<tr>
<td>vrf-name</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-if)# vrf vrf1</td>
</tr>
<tr>
<td>Step 4 ipv4 address</td>
<td>Assigns an IP address and subnet mask to the interface.</td>
</tr>
<tr>
<td>ip-address mask</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-if)# ipv4 address 172.18.189.38 255.255.255.225</td>
</tr>
<tr>
<td>Step 5 end</td>
<td>Saves configuration changes. When you issue the end command, the system prompts you to commit changes:</td>
</tr>
<tr>
<td>Example:</td>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| **Purpose**      | • Entering `yes` saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.  
   • Entering `no` exits the configuration session and returns the router to EXEC mode without committing the configuration changes.  
   • Entering `cancel` leaves the router in the current configuration session without exiting or committing the configuration changes. |
| **Step 6**       | Enables OSPF routing for the specified routing process, and places the router in router configuration mode. In this example, the OSPF instance is called `isp`. |
| `router ospf`    |        |
| `instance-id`    |        |
| `Example:`       |        |
| `RP/0/RSP0/CPU0:router(config)# router ospf isp` | |
| **Step 7**       | Creates a VRF instance and enters VRF configuration mode. |
| `vrf`            |        |
| `vrf-name`       |        |
| `Example:`       |        |
| `RP/0/RSP0/CPU0:router(config-ospf)# vrf vrf1` | |
| **Step 8**       | Configures a router ID for the OSPF process. |
| `router-id`      |        |
| `{ router-id }`  |        |
| `Example:`       |        |
| `RP/0/RSP0/CPU0:router(config-ospf-vrf)# router-id 192.168.4.3` | |
| **Step 9**       | Redistributes OSPF routes from the one routing domain to another routing domain. |
| `redistribute bgp` |        |
| `process-id`     |        |
| `Example:`       |        |
| `RP/0/RSP0/CPU0:router(config-ospf-vrf)# redistribute bgp 1` | |
| **Step 10**      | Enters area configuration mode and configures an area for the OSPF process. |
| `area`           |        |
| `area-id`        |        |
| `Example:`       |        |
| `RP/0/RSP0/CPU0:router(config-ospf-vrf)# area 0` | |

*Note:* We recommend using a stable IPv4 address as the router ID.
### 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/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ospfv3)# nsr
Enable NSR for the OSPFv3 process.

Step 4 commit
Commit your configuration.

Configuring OSPF SPF Prefix Prioritization

Perform this task to configure OSPF SPF (shortest path first) prefix prioritization.

SUMMARY STEPS

1. configure
2. prefix-set prefix-set name
3. route-policy route-policy name if destination in prefix-set name then set spf-priority {critical | high | medium} endif
4. Use one of these commands:
   • router ospf ospf-name
   • router ospfv3 ospfv3-name
5. spf prefix-priority route-policy route-policy name
6. commit
7. show rpl route-policy route-policy name detail
# Configuring OSPF SPF Prefix Prioritization

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><strong>configure</strong></td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><strong>prefix-set</strong> <em>prefix-set name</em></td>
<td>Configures the prefix set.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RP/0/RSP0/CPU0:router(config)#prefix-set ospf-critical-prefixes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RP/0/RSP0/CPU0:router(config-pfx)#66.0.0.0/16</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RP/0/RSP0/CPU0:router(config-pfx)#end-set</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><strong>route-policy</strong> <em>route-policy name</em> <em>if destination in</em> <em>prefix-set name</em> <em>then</em> <em>set spf-priority</em> *{critical</td>
<td>high</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RP/0/RSP0/CPU0:router#route-policy ospf-spf-priority</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RP/0/RSP0/CPU0:router(config-rpl)#if destination in ospf-critical-prefixes then</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>set spf-priority critical</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>endif</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RP/0/RSP0/CPU0:router(config-rpl)#end-policy</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>Use one of these commands:</td>
<td>Enters Router OSPF configuration mode.</td>
</tr>
<tr>
<td></td>
<td>• <strong>router ospf ospf-name</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>router ospfv3 ospfv3-name</strong></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RP/0/RSP0/CPU0:router# router ospf 1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Or</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RP/0/RSP0/CPU0:router# router ospfv3 1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td><strong>spf prefix-priority route-policy</strong> <em>route-policy name</em></td>
<td>Configures SPF prefix-priority for the defined route policy.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Or</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RP/0/RSP0/CPU0:router(config-ospfv3)#spf prefix-priority</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>route-policy ospfv3-spf-priority</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>Configure the <strong>spf prefix-priority</strong> command under router OSPF.</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td><strong>commit</strong></td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td><strong>show rpl route-policy</strong> <em>route-policy name</em> <em>detail</em>*</td>
<td>Displays the set SPF prefix priority.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RP/0/RSP0/CPU0:router#show rpl route-policy ospf-spf-priority detail</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>prefix-set ospf-critical-prefixes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>66.0.0.0/16</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>end-set</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>route-policy ospf-spf-priority</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Enabling Multicast-intact for OSPFv2

This optional task describes how to enable multicast-intact for OSPFv2 routes that use IPv4 addresses.

#### SUMMARY STEPS

1. configure
2. router ospf instance-id
3. mpls traffic-eng multicast-intact
4. commit

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router ospf instance-id</td>
<td>Enables OSPF routing for the specified routing process, and places the router in router configuration mode. In this example, the OSPF instance is called isp.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router ospf isp</td>
<td></td>
</tr>
<tr>
<td>Step 3 mpls traffic-eng multicast-intact</td>
<td>Enables multicast-intact.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-ospf)# mpls traffic-eng multicast-intact</td>
<td></td>
</tr>
<tr>
<td>Step 4 commit</td>
<td></td>
</tr>
</tbody>
</table>

### Associating Interfaces to a VRF

This task explains how to associate an interface with a VPN Routing and Forwarding (VRF) instance.
### SUMMARY STEPS

1. `configure`
2. `router ospf process-name`
3. `vrf vrf-name`
4. `area area-id`
5. `interface type interface-path-id`
6. `commit`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>configure</code></td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><code>router ospf process-name</code></td>
<td>Enables OSPF routing for the specified routing process and places the</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>router in router configuration mode.</td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config)# router ospf 1</code></td>
<td>Note: The <code>process-name</code> argument is any alphanumeric string no longer than 40 characters.</td>
</tr>
<tr>
<td>Step 3</td>
<td><code>vrf vrf-name</code></td>
<td>Creates a VRF instance and enters VRF configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-ospf)# vrf vrf1</code></td>
<td>END</td>
</tr>
<tr>
<td>Step 4</td>
<td><code>area area-id</code></td>
<td>Enters area configuration mode and configures an area for the OSPF</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>process.</td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-ospf-vrf)# area 0</code></td>
<td>• The <code>area-id</code> 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.</td>
</tr>
<tr>
<td>Step 5</td>
<td><code>interface type interface-path-id</code></td>
<td>Enters interface configuration mode and associates one or more interfaces to the VRF.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-ospf-vrf-ar)# interface GigabitEthernet 0/0/0</code></td>
<td>END</td>
</tr>
<tr>
<td>Step 6</td>
<td><code>commit</code></td>
<td></td>
</tr>
</tbody>
</table>
Configuring OSPF as a Provider Edge to Customer Edge (PE-CE) Protocol

SUMMARY STEPS

1. configure
2. router ospf process-name
3. vrf vrf-name
4. router-id { router-id }
5. redistribute protocol [ process-id ] [ level-1 | level-1-2 | level-2 ] [ metric metric-value ] [ metric-type type-value ] [ match { external [ 1 | 2 ] } ] [ tag tag-value ] route-policy policy-name
6. area area-id
7. interface type interface-path-id
8. exit
9. domain-id / secondary / type / 0005 | 0105 | 0205 | 8005 / value value
10. domain-tag tag
11. disable-dn-bit-check
12. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router ospf process-name</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
</tbody>
</table>
| **Example:**  
RP/0/RSP0/CPU0:router(config)# router ospf 1 | The process-name argument is any alphanumeric string no longer than 40 characters. |
| **Step 3** vrf vrf-name | Creates a VRF instance and enters VRF configuration mode. |
| **Example:**  
RP/0/RSP0/CPU0:router(config-ospf)# vrf vrf1 | |
| **Step 4** router-id { router-id } | Configures a router ID for the OSPF process. |
| **Note** We recommend using a stable IPv4 address as the router ID. |
| **Example:**  
RP/0/RSP0/CPU0:router(config-ospf-vrf)# router-id 192.168.4.3 | |
<p>| <strong>Step 5</strong> 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 | Redistributes OSPF routes from one routing domain to another routing domain. |
| Redistributes OSPF routes from one routing domain to another routing domain. |
| • This command causes the router to become an ASBR by definition. |</p>
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Example:** RP/0/RSP0/CPU0:router(config-ospf-vrf)# redistribute bgp 1 level-1 | • OSPF tags all routes learned through redistribution as external.  
• The protocol and its process ID, if it has one, indicate the protocol being redistributed into OSPF.  
• The metric is the cost you assign to the external route. The default is 20 for all protocols except BGP, whose default metric is 1.  
• The example shows the redistribution of BGP autonomous system 1, Level 1 routes into OSPF as Type 2 external routes. |
| **Step 6** area area-id | Enters area configuration mode and configures an area for the OSPF process.  
| Example: RP/0/RSP0/CPU0:router(config-ospf-vrf)# area 0 | • The area-id argument can be entered in dotted-decimal or IPv4 address notation, such as area1000 or area 0.0.3.232. However, you must choose one form or the other for an area. |
| **Step 7** interface type interface-path-id | Enters interface configuration mode and associates one or more interfaces to the VRF. |
| Example: RP/0/RSP0/CPU0:router(config-ospf-vrf)# interface GigabitEthernet 0/0/0/0 | |
| **Step 8** exit | Exits interface configuration mode. |
| Example: RP/0/RSP0/CPU0:router(config-if)# exit | |
| **Step 9** domain-id / secondary / type / 0005 | Specifies the OSPF VRF domain ID.  
| Example: RP/0/RSP0/CPU0:router(config-ospf-vrf)# domain-id type 0105 value 1AF234 | • The value argument is a six-octet hex number. |
| | value 8005 | |
| | value 0005 | |
| **Step 10** domain-tag tag | Specifies the OSPF VRF domain tag.  
| Example: RP/0/RSP0/CPU0:router(config-ospf-vrf)# domain-tag 234 | • The valid range for tag is 0 to 4294967295. |
Creating Multiple OSPF Instances (OSPF Process and a VRF)

This task explains how to create multiple OSPF instances. In this case, the instances are a normal OSPF instance and a VRF instance.

**SUMMARY STEPS**

1. configure
2. `router ospf  process-name`
3. `area  area-id`
4. `interface  type  interface-path-id`
5. `exit`
6. `vrf  vrf-name`
7. `area  area-id`
8. `interface  type  interface-path-id`
9. `commit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>router ospf  process-name</code></td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode. <strong>Note</strong> The <code>process-name</code> argument is any alphanumeric string no longer than 40 characters.</td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>area  area-id</code></td>
<td>Enters area configuration mode and configures a backbone area. <strong>•</strong> The <code>area-id</code> 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.</td>
</tr>
</tbody>
</table>

---

### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 11</strong> disable-dn-bit-check</td>
<td>Specifies that down bits should be ignored.</td>
</tr>
</tbody>
</table>

**Example:**

```bash
RP/0/RSP0/CPU0:router(config-ospf-vrf)# disable-dn-bit-check
```
### Configuring Multi-area Adjacency

This task explains how to create multiple areas on an OSPF primary interface.

**Before You Begin**

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.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 4</strong></td>
<td>Enters interface configuration mode and associates one or more interfaces to the area.</td>
</tr>
</tbody>
</table>
| **interface type interface-path-id** | **Example:**  
RP/0/RSP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet 0/1/0/3 |
| **Step 5** | Enters OSPF configuration mode. |
| **exit** | **Example:**  
RP/0/RSP0/CPU0:router(config-ospf-ar)# exit |
| **Step 6** | Creates a VRF instance and enters VRF configuration mode. |
| **vrf vrf-name** | **Example:**  
RP/0/RSP0/CPU0:router(config-ospf-ar)# vrf vrf1 |
| **Step 7** | Enters area configuration mode and configures an area for a VRF instance under 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. |
| **area area-id** | **Example:**  
RP/0/RSP0/CPU0:router(config-ospf-vrf)# area 0 |
| **Step 8** | Enters interface configuration mode and associates one or more interfaces to the VRF. |
| **interface type interface-path-id** | **Example:**  
RP/0/RSP0/CPU0:router(config-ospf-vrf)# interface GigabitEthernet 0/0/0/0 |
| **Step 9** | commit |

**Configuring Multi-area Adjacency**

Before You Begin

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. `commit`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router ospf <code>process-name</code></td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# router ospf 1</td>
<td>The <code>process-name</code> argument is any alphanumeric string no longer than 40 characters.</td>
</tr>
<tr>
<td><strong>Step 3</strong> area <code>area-id</code></td>
<td>Enters area configuration mode and configures a backbone area.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-ospf)# area 0</td>
<td>- The <code>area-id</code> 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.</td>
</tr>
<tr>
<td><strong>Step 4</strong> interface <code>type interface-path-id</code></td>
<td>Enters interface configuration mode and associates one or more interfaces to the area.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-ospf-ar)# interface Serial 0/1/0/3</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> area <code>area-id</code></td>
<td>Enters area configuration mode and configures an area used for multiple area adjacency.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-ospf)# area 1</td>
<td>- The <code>area-id</code> 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.</td>
</tr>
</tbody>
</table>
### Configuring Label Distribution Protocol IGP Auto-configuration for OSPF

This task explains how to configure LDP auto-configuration for an OSPF instance. Optionally, you can configure this feature for an area of an OSPF instance.

#### SUMMARY STEPS

1. configure
2. router ospf process-name
3. mpls ldp auto-config
4. commit

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>configure</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>router ospf</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td></td>
<td>process-name</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config)# router ospf 1
```

**Note**  
The process-name argument is any alphanumeric string no longer than 40 characters.

<table>
<thead>
<tr>
<th>Step 3</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mpls ldp auto-config</td>
<td>Enables LDP IGP interface auto-configuration for an OSPF instance.</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config-ospf)# mpls ldp auto-config
```

- Optionally, this command can be configured for an area of an OSPF instance.

<table>
<thead>
<tr>
<th>Step 4</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
</tr>
<tr>
<td>Step 2</td>
<td>router ospf process-name</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router ospf 100</td>
</tr>
<tr>
<td>Step 3</td>
<td>Use one of the following commands:</td>
</tr>
<tr>
<td></td>
<td>• mpls ldp sync</td>
</tr>
<tr>
<td></td>
<td>• area area-id mpls ldp sync</td>
</tr>
<tr>
<td></td>
<td>• area area-id interface name mpls ldp sync</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf)# mpls ldp sync</td>
</tr>
<tr>
<td>Step 4</td>
<td>commit</td>
</tr>
</tbody>
</table>
Configuring Authentication Message Digest Management for OSPF

This task explains how to manage authentication of a keychain on the OSPF interface.

Before You Begin
A valid keychain must be configured before this task can be attempted.

To learn how to configure a keychain and its associated attributes, see the Implementing Key Chain Management on Cisco ASR 9000 Series Router module of the Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide.

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  keychain
7. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
</tbody>
</table>
| **Step 2** router ospf  process-name | **Example:**
RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ospf)# area 1 |
<p>| | 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. |</p>
<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>interface type interface-path-id</td>
<td>Enters interface configuration mode and associates one or more interfaces to the area.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet0/4/0/1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>authentication message-digest keychain keychain</td>
<td>Configures an MD5 keychain.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar-if)# authentication message-digest keychain ospf_int1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note</td>
<td>In the example, the ospf_int1 keychain must be configured before you attempt this step.</td>
</tr>
<tr>
<td>7</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

### 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.

```plaintext
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:

```plaintext
show ospf 1 interface GigabitEthernet0/4/0/1
```
Configuring Generalized TTL Security Mechanism (GTSM) for OSPF

This task explains how to set the security time-to-live mechanism on an interface for GTSM.

**SUMMARY STEPS**

1. configure
2. router ospf process-name
3. router-id { router-id }
4. log adjacency changes [ detail | disable ]
5. nsf { cisco [ enforce global ] | ietf [ helper disable ] }
6. timers throttle spf spf-start spf-hold spf-max-wait
7. area area-id
8. interface type interface-path-id
9. security ttl [ disable | hops hop-count ]
10. commit
11. show ospf [ process-name ] [ vrf vrf-name ] [ area-id ] interface [ type interface-path-id ]
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td><strong>Purpose</strong> Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router ospf  process-name</td>
<td>The <code>process-name</code> argument is any alphanumeric string no longer than 40 characters.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config)# router ospf 1</td>
</tr>
<tr>
<td><strong>Step 3</strong> router-id { router-id }</td>
<td>Configures a router ID for the OSPF process.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>We recommend using a stable IPv4 address as the router ID.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf)# router id 10.10.10.100</td>
</tr>
<tr>
<td><strong>Step 4</strong> log adjacency changes [ detail</td>
<td>disable ]</td>
</tr>
<tr>
<td></td>
<td>- By default, this feature is enabled.</td>
</tr>
<tr>
<td></td>
<td>- The messages generated by neighbor changes are considered notifications, which are categorized as severity Level 5 in the <code>logging console</code> command. The <code>logging console</code> command controls which severity level of messages are sent to the console. By default, all severity level messages are sent.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar-if)# log adjacency changes detail</td>
</tr>
<tr>
<td><strong>Step 5</strong> nsf { cisco</td>
<td>[ enforce global ]</td>
</tr>
<tr>
<td></td>
<td>The example enables graceful restart.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>We recommend using the IPv4 address notation.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf)# nsf ietf</td>
</tr>
<tr>
<td><strong>Step 6</strong> timers throttle spf <code>spf-start spf-hold spf-max-wait</code></td>
<td>(Optional) Sets SPF throttling timers.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf)# timers throttle spf 500 500 10000</td>
</tr>
<tr>
<td><strong>Step 7</strong> area area-id</td>
<td>Enters area configuration mode.</td>
</tr>
<tr>
<td></td>
<td>The <code>area-id</code> 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.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf)# area 1</td>
</tr>
<tr>
<td><strong>Step 8</strong> interface type interface-path-id</td>
<td>Enters interface configuration mode and associates one or more interfaces to the area.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet0/5/0/0</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>security ttl [ disable hops hop-count ]</td>
</tr>
<tr>
<td>Example:</td>
<td>RP0/RSP0/CPU0:router(config-ospf-ar-if)#security ttl hop-count 2</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Sets the security TTL value in the IP header for OSPF packets.</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>commit</td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>show ospf [ process-name ] [ vrf vrf-name ] [ area-id ] interface [ type interface-path-id ]</td>
</tr>
<tr>
<td>Example:</td>
<td>RP0/RSP0/CPU0:router# show ospf 1 interface GigabitEthernet0/5/0/0</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Displays OSPF interface information.</td>
</tr>
</tbody>
</table>

### 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 120.10.10.24, Area 0
Process ID 1, Router ID 100.100.100.100, Network Type BROADCAST, Cost: 1
Transmit Delay is 1 sec, State BDR, Priority 1
TTL security enabled, hop count 2
Designated Router (ID) 102.102.102.102, Interface address 120.10.10.3
Backup Designated router (ID) 100.100.100.100, Interface address 120.10.10.1
Flush timer for old DR LSA due in 00:02:36
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:05
Index 1/1, flood queue length 0
Next 0/0/0/0
Last flood scan length is 1, maximum is 4
Last flood scan time is 0 msec, maximum is 0 msec
Neighbor Count is 1, Adjacent neighbor count is 1
Adjacent with neighbor 102.102.102.102 (Designated Router)
Suppress hello for 0 neighbor(s)
Multi-area interface Count is 0
```

### Verifying OSPF Configuration and Operation

This task explains how to verify the configuration and operation of OSPF.
### SUMMARY STEPS

1. `show { ospf | ospfv3 } [ process-name ]`
2. `show { ospf | ospfv3 } [ process-name ] border-routers [ router-id ]`
3. `show { ospf | ospfv3 } [ process-name ] database`
4. `show { ospf | ospfv3 } [ process-name ] [ area-id ] flood-list interface type interface-path-id`
5. `show { ospf | ospfv3 } [ process-name ] [ vrf vrf-name ] [ area-id ] interface [ type interface-path-id ]`
6. `show { ospf | ospfv3 } [ process-name ] [ area-id ] neighbor [ type interface-path-id ] [ neighbor-id ] [ detail ]`
7. `clear { ospf | ospfv3 } [ process-name ] process`
8. `clear { ospf | ospfv3 } [ process-name ] redistribution`
9. `clear { ospf | ospfv3 } [ process-name ] routes`
10. `clear { ospf | ospfv3 } [ process-name ] vrf [ vrf-name | all ] [ process | redistribution | routes | statistics ] [ interface type interface-path-id | message-queue | neighbor ]`
11. `clear { ospf | ospfv3 } [ process-name ] statistics [ neighbor [ type interface-path-id ] [ ip-address ] ]`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>`show { ospf</td>
<td>ospfv3 } [ process-name ]`</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router# show ospf group1</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>`show { ospf</td>
<td>ospfv3 } [ process-name ] border-routers [ router-id ]`</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router# show ospf group1 border-routers</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>`show { ospf</td>
<td>ospfv3 } [ process-name ] database`</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router# show ospf group2 database</td>
<td>• The various forms of this command deliver information about different OSPF LSAs.</td>
</tr>
<tr>
<td>Step 4</td>
<td>`show { ospf</td>
<td>ospfv3 } [ process-name ] [ area-id ] flood-list interface type interface-path-id`</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router# show ospf 100 flood-list interface GigabitEthernet 0/3/0/0</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>5</td>
<td>`show { ospf</td>
<td>ospfv3 } [ process-name ] [ vrf vrf-name ] [ area-id ] interface [ type interface-path-id ]`</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router# show ospf 100 interface GigabitEthernet 0/3/0/0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>`show { ospf</td>
<td>ospfv3 } [{ process-name } [ area-id ] neighbor [ type interface-path-id ] [ neighbor-id ] [ detail ]`</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router# show ospf 100 neighbor</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>`clear { ospf</td>
<td>ospfv3 } [{ process-name } process]`</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router# clear ospf 100 process</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td><code>clear ospfv3 [ process-name ] redistribution</code></td>
<td>Clears OSPF route redistribution.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router# clear ospf 100 redistribution</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td><code>clear ospfv3 [ process-name ] routes</code></td>
<td>Clears OSPF route table.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router# clear ospf 100 routes</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router# clear ospf 100 vrf vrf_1 process</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>`clear { ospf</td>
<td>ospfv3 } [{ process-name } statistics [ neighbor [ type interface-path-id ] [ ip-address ] ]`</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router# clear ospf 100 statistics</td>
<td></td>
</tr>
</tbody>
</table>
Configuring IP Fast Reroute Loop-free Alternate

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

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

Enabling IPFRR LFA

**SUMMARY STEPS**

1. configure
2. router ospf process-name
3. area area-id
4. interface type interface-path-id
5. fast-reroute per-link { enable | disable }
6. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Step 2</td>
<td>router ospf process-name</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router ospf</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>area area-id</td>
<td>Enters area configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf)#area 1</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>interface type interface-path-id</td>
<td>Enters interface configuration mode and associates one or more interfaces to the area.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)#interface GigabitEthernet0/5/0/0</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>fast-reroute per-link { enable</td>
<td>disable }</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)#fast-reroute per-link enable</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>
Excluding an Interface From IP Fast Reroute Per-link Computation

SUMMARY STEPS

1. configure
2. router ospf  process-name
3. area  area-id
4. interface  type interface-path-id
5. fast-reroute per-link exclude interface  type interface-path-id
6. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>router ospf  process-name</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router ospf</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>area  area-id</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)#area area-id</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>interface  type interface-path-id</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf)#interface type interface-path-id</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>fast-reroute per-link exclude interface  type interface-path-id</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# fast-reroute per-link exclude interface GigabitEthernet0/5/0/1</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>commit</td>
</tr>
</tbody>
</table>

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-map advertise-local`
6. `segment-routing sr-prefer prefix-list [acl-name]`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>configure</code></td>
<td>Enables OSPF routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Step 2</td>
<td><code>router ospf instance-id</code></td>
<td>Enables OSPF routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config)# router ospf isp</code></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>segment-routing mpls</code></td>
<td>Enables SR forwarding on all interfaces where this instance OSPF is enabled.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-ospf)# segment-routing mpls</code></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>segment-routing forwarding mpls</code></td>
<td>Enables SR forwarding on all interfaces where this instance OSPF is enabled.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-ospf)# segment-routing forwarding mpls</code></td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td><code>segment-routing prefix-sid-map advertise-local</code></td>
<td>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.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-ospf)# segment-routing prefix-sid-map advertise local</code></td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td><code>segment-routing sr-prefer prefix-list [acl-name]</code></td>
<td>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.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-ospf)# segment-routing sr-prefer prefix-list foo</code></td>
<td></td>
</tr>
</tbody>
</table>

The following example shows how OSPF advertises local mapping entries using area-flooding scope.

```
ipv4 prefix-list foo
permit 2.2.2.2/32
```
**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**

```plaintext
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**

```plaintext
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**

```plaintext
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.

```plaintext
router ospf 1
  router-id 10.5.4.3
  cost 5
  area 0
    interface GigabitEthernet 0/1/0/0

area 1
  interface GigabitEthernet 0/2/0/0

area 6
  interface GigabitEthernet 0/3/0/0
```
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 Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide for information.

In this example, loopback interface 0 is associated with area 0 and MPLS TE is configured within area 0.

```
interface Loopback 0
description 10.10.10.10 255.255.252.0
interface GigabitEthernet 0/2/0/0
description 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:

```conf
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:

```conf
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:

```conf
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
```
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**

```conf
cisco
routerr 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**

```conf
cisco
routerr 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
```

---

**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 301.

In this example, all interfaces on router ABR1 use MD5 authentication:

```conf
cisco
routerr ospfv3 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:

```conf
cisco
routerr ospfv3 ABR2
router-id 10.10.5.5
area 0
  area 1
  authentication message-digest
  message-digest-key 100 md5 0 cisco
  interface GigabitEthernet 0/9/0/1
  virtual-link 10.10.10.10
  area 3
  interface Loopback 0
  interface GigabitEthernet 0/9/0/0
!```
VPN Backbone and Sham Link Configured for OSPF Version 2: Example

The following examples show how to configure a provider edge (PE) router to establish a VPN backbone and sham link connection:

```
logging console debugging
vrf vrf_1
  address-family ipv4 unicast
  import route-target 100:1
  export route-target 100:1

interface Loopback0
  ipv4 address 2.2.2.1 255.255.255.255

interface Loopback1
  vrf vrf_1
  ipv4 address 10.0.1.3 255.255.255.255

interface GigabitEthernet0/2/0/2
  vrf vrf_1
  ipv4 address 100.10.10.2 255.255.255.0

interface GigabitEthernet0/2/0/3
  ipv4 address 100.20.10.2 255.255.255.0

route-policy pass-all
  pass
  end-policy

router ospf 1
  log adjacency changes
  router-id 2.2.2.2
  vrf vrf_1
  router-id 22.22.22.2
  domain-id type 0005 value 1111222333
  domain-tag 140
  nsf ietf
  redistribute bgp 10
  area 0
    sham-link 10.0.1.3 10.0.0.101
    ! interface GigabitEthernet0/2/0/2
    !
    !
  router ospf 2
    router-id 2.22.2.22
    area 0
    interface Loopback0
    !
    !
  router bgp 10
    bgp router-id 2.2.2.1
    bgp graceful-restart restart-time 300
    bgp graceful-restart
    address-family ipv4 unicast
    redistribute connected
    !
    address-family vpnv4 unicast
```
Where to Go Next

To configure route maps through the RPL for OSPF Version 2, see Implementing Routing Policy on Cisco ASR 9000 Series Router module.

To build an MPLS TE topology, create tunnels, and configure forwarding over the tunnel for OSPF Version 2; see Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide.

Additional References

The following sections provide references related to implementing OSPF.

Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSPF Commands and OSPFv3 Commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td>Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference</td>
</tr>
<tr>
<td>MPLS TE feature information</td>
<td>Implementing MPLS Traffic Engineering on Cisco ASR 9000 Series Router module in Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide</td>
</tr>
<tr>
<td>MIB Reference</td>
<td>Cisco ASR 9000 Series Aggregation Services Router MIB Specification Guide</td>
</tr>
</tbody>
</table>
## Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>draft-ietf-ospf-multi-area-adj-07.txt</td>
<td>OSPF Multi-Area Adjacency</td>
</tr>
<tr>
<td>draft-ietf-pce-disco-proto-ospf-08.txt</td>
<td>OSPF Protocol Extensions for Path Computation Element (PCE)</td>
</tr>
<tr>
<td>draft-ietf-mpls-igp-sync-00.txt</td>
<td>LDP IGP Synchronization</td>
</tr>
<tr>
<td>draft-ietf-ospf-ospfv3-graceful-restart-07.txt</td>
<td>OSPFv3 Graceful Restart</td>
</tr>
</tbody>
</table>

## MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>To locate and download MIBs for selected platforms, Cisco IOS releases,</td>
</tr>
<tr>
<td></td>
<td>and feature sets, use Cisco MIB Locator found at the following URL:</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.cisco.com/go/mibs">http://www.cisco.com/go/mibs</a></td>
</tr>
</tbody>
</table>

## RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 1587</td>
<td>The OSPF NSSA Option</td>
</tr>
<tr>
<td>RFC 1793</td>
<td>Extending OSPF to Support Demand Circuits</td>
</tr>
<tr>
<td>RFC 2328</td>
<td>OSPF Version 2</td>
</tr>
<tr>
<td>RFC 2370</td>
<td>The OSPF Opaque LSA Option</td>
</tr>
<tr>
<td>RFC 2740</td>
<td>OSPF for IPv6</td>
</tr>
<tr>
<td>RFC 3101</td>
<td>The OSPF Not-So-Stubby Area (NSSA) Option</td>
</tr>
<tr>
<td>RFC 3137</td>
<td>OSPF Stub Router Advertisement</td>
</tr>
<tr>
<td>RFC 3509</td>
<td>Alternative Implementations of OSPF Area Border Routers</td>
</tr>
<tr>
<td>RFC 3623</td>
<td>Graceful OSPF Restart</td>
</tr>
<tr>
<td>RFC 3630</td>
<td>Traffic Engineering (TE) Extensions to OSPF Version 2</td>
</tr>
</tbody>
</table>
### RFCs

<table>
<thead>
<tr>
<th>RFC</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 3682</td>
<td>The Generalized TTL Security Mechanism (GTSM)</td>
</tr>
<tr>
<td>RFC 3906</td>
<td>Calculating Interior Gateway Protocol (IGP) Routes Over Traffic Engineering Tunnels</td>
</tr>
<tr>
<td>RFC 4136</td>
<td>OSPF Refresh and Flooding Reduction in Stable Topologies</td>
</tr>
<tr>
<td>RFC 4206</td>
<td>Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)</td>
</tr>
<tr>
<td>RFC 4124</td>
<td>Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering</td>
</tr>
<tr>
<td>RFC 4576</td>
<td>Using a Link State Advertisement (LSA) Options Bit to Prevent Looping in BGP/MPLS IP Virtual Private Networks (VPNs) ownbit Extension for L3VPN</td>
</tr>
<tr>
<td>RFC 4577</td>
<td>OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)</td>
</tr>
<tr>
<td>RFC 4750</td>
<td>OSPF Version 2 Management Information Base</td>
</tr>
<tr>
<td>RFC 4811</td>
<td>OSPF Out-of-Band Link State Database (LSDB) Resynchronization</td>
</tr>
<tr>
<td>RFC 4812</td>
<td>OSPF Restart Signaling</td>
</tr>
<tr>
<td>RFC 4813</td>
<td>OSPF Link-Local Signaling</td>
</tr>
<tr>
<td>RFC 4970</td>
<td>Extensions to OSPF for Advertising Optional Router Capabilities</td>
</tr>
<tr>
<td>RFC 5643</td>
<td>Management Information Base (MIB) for OSPFv3</td>
</tr>
</tbody>
</table>

### Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a></td>
</tr>
</tbody>
</table>
CHAPTER 7

Implementing and Monitoring RIB

Routing Information Base (RIB) is a distributed collection of information about routing connectivity among all nodes of a network. Each router maintains a RIB containing the routing information for that router. RIB stores the best routes from all routing protocols that are running on the system.

This module describes how to implement and monitor RIB on Cisco IOS XR network.

Note

For more information about RIB on the Cisco IOS XR software and complete descriptions of RIB commands listed in this module, see the Additional References section of this module.

To locate documentation for other commands that might appear during the execution of a configuration task, search online in the Cisco ASR 9000 Series Aggregation Services Router Commands Master List.

Feature History for Implementing and Monitoring RIB

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.7.2</td>
<td>This feature was introduced.</td>
</tr>
<tr>
<td>Release 4.2.0</td>
<td>The following features were added:</td>
</tr>
<tr>
<td></td>
<td>• Route and Label Consistency Checker (RCC and LCC)</td>
</tr>
<tr>
<td></td>
<td>• System-wide Route Prioritization for IOS XR Software</td>
</tr>
<tr>
<td>Release 4.2.1</td>
<td>BGP Prefix Independent Convergence for RIB and FIB support was added.</td>
</tr>
</tbody>
</table>

- Prerequisites for Implementing RIB, page 382
- Information About RIB Configuration, page 382
- How to Deploy and Monitor RIB, page 386
- Configuring RCC and LCC, page 390
- Configuration Examples for RIB Monitoring, page 392
Prerequisites for Implementing RIB

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

- RIB is distributed with the base Cisco IOS XR software; as such, it does not have any special requirements for installation. The following are the requirements for base software installation:
  - Router
  - Cisco IOS XR software
  - Base package

Information About RIB Configuration

To implement the Cisco RIB feature, you must understand the following concepts:

Overview of RIB

Each routing protocol selects its own set of best routes and installs those routes and their attributes in RIB. RIB stores these routes and selects the best ones from among all routing protocols. Those routes are downloaded to the line cards for use in forwarding packets. The acronym RIB is used both to refer to RIB processes and the collection of route data contained within RIB.

Within a protocol, routes are selected based on the metrics in use by that protocol. A protocol downloads its best routes (lowest or tied metric) to RIB. RIB selects the best overall route by comparing the administrative distance of the associated protocol.

RIB Data Structures in BGP and Other Protocols

RIB uses processes and maintains data structures distinct from other routing applications, such as Border Gateway Protocol (BGP) and other unicast routing protocols, or multicast protocols, such as Protocol Independent Multicast (PIM) or Multicast Source Discovery Protocol (MSDP). However, these routing protocols use internal data structures similar to what RIB uses, and may internally refer to the data structures as a RIB. For example, BGP routes are stored in the BGP RIB (BRIB), and multicast routes, computed by multicast routing protocols such as PIM and MSDP, are stored in the Multicast RIB (MRIB). RIB processes are not responsible for the BRIB and MRIB, which are handled by BGP and multicast processes, respectively.

The table used by the line cards and RP to forward packets is called the Forwarding Information Base (FIB). RIB processes do not build the FIBs. Instead, RIB downloads the set of selected best routes to the FIB processes, by the Bulk Content Downloader (BCDL) process, onto each line card. FIBs are then constructed.
RIB Administrative Distance

Forwarding is done based on the longest prefix match. If you are forwarding a packet destined to 10.0.2.1, you prefer 10.0.2.0/24 over 10.0.0.0/16 because the mask /24 is longer (and more specific) than a /16.

Routes from different protocols that have the same prefix and length are chosen based on administrative distance. For instance, the Open Shortest Path First (OSPF) protocol has an administrative distance of 110, and the Intermediate System-to-Intermediate System (IS-IS) protocol has an administrative distance of 115. If IS-IS and OSPF both download 10.0.1.0/24 to RIB, RIB would prefer the OSPF route because OSPF has a lower administrative distance. Administrative distance is used only to choose between multiple routes of the same length.

This table lists default administrative distances for the common protocols.

Table 3: Default Administrative Distances

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Administrative Distance Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected or local routes</td>
<td>0</td>
</tr>
<tr>
<td>Static routes</td>
<td>1</td>
</tr>
<tr>
<td>External BGP routes</td>
<td>20</td>
</tr>
<tr>
<td>OSPF routes</td>
<td>110</td>
</tr>
<tr>
<td>IS-IS routes</td>
<td>115</td>
</tr>
<tr>
<td>Internal BGP routes</td>
<td>200</td>
</tr>
</tbody>
</table>

The administrative distance for some routing protocols (for instance IS-IS, OSPF, and BGP) can be changed. See the protocol-specific documentation for the proper method to change the administrative distance of that protocol.

**Note** Changing the administrative distance of a protocol on some but not all routers can lead to routing loops and other undesirable behavior. Doing so is not recommended.

RIB Support for IPv4 and IPv6

In Cisco IOS XR software, RIB tables support multicast and unicast routing.

The default routing tables for Cisco IOS XR software RIB are the unicast RIB tables for IPv4 routing and the multicast-unicast RIB tables for IPv6 routing. For multicast routing, routing protocols insert unicast routes into the multicast-unicast RIB table. Multicast protocols then use the information to build multicast routes (which in turn are stored in the MRIB). See the multicast documentation for more information on using and configuring multicast.
RIB processes ipv4_rib and ipv6_rib run on the RP card. If process placement functionality is available and supported by multiple RPs in the router, RIB processes can be placed on any available node.

**RIB Statistics**

RIB supports statistics for messages (requests) flowing between the RIB and its clients. Protocol clients send messages to the RIB (for example, route add, route delete, and next-hop register, and so on). RIB also sends messages (for example, redistribute routes, advertisements, next-hop notifications, and so on). These statistics are used to gather information about what messages have been sent and the number of messages that have been sent. These statistics provide counters for the various messages that flow between the RIB server and its clients. The statistics are displayed using the `show rib statistics` command.

RIB maintains counters for all requests sent from a client including:

- Route operations
- Table registrations
- Next-hop registrations
- Redistribution registrations
- Attribute registrations
- Synchronization completion

RIB also maintains counters for all requests sent by the RIB. The configuration will disable the RIB next-hop dampening feature. As a result, RIB notifies client immediately when a next hop that client registered for is resolved or unresolved.

RIB also maintains the results of the requests.

**IPv6 Provider Edge IPv6 and IPv6 VPN Provider Edge Transport over MPLS**

IPv6 Provider Edge (6PE) and IPv6 VPN Provider Edge (6VPE) leverages the existing Multiprotocol Label Switching (MPLS) IPv4 core infrastructure for IPv6 transport. 6PE and 6VPE enable IPv6 sites to communicate with each other over an MPLS IPv4 core network using MPLS label switched paths (LSPs).

RIB supports 6PE and 6VPE by providing 6VPE next hops. The next-hop information is stored in an opaque database in RIB, which is populated by protocol clients with data to be sent to the Forwarding Information Base (FIB).

For detailed information about configuring 6PE and 6VPE over MPLS, see *Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide*.

**RIB Quarantining**

RIB quarantining solves the problem in the interaction between routing protocols and the RIB. The problem is a persistent oscillation between the RIB and routing protocols that occurs when a route is continuously inserted and then withdrawn from the RIB, resulting in a spike in CPU use until the problem is resolved. If there is no damping on the oscillation, then both the protocol process and the RIB process have high CPU use, affecting the rest of the system as well as blocking out other protocol and RIB operations. This problem occurs when a particular combination of routes is received and installed in the RIB. This problem typically
happens as a result of a network misconfiguration. However, because the misconfiguration is across the
network, it is not possible to detect the problem at configuration time on any single router.

The quarantining mechanism detects mutually recursive routes and quarantines the last route that completes
the mutual recursion. The quarantined route is periodically evaluated to see if the mutual recursion has gone
away. If the recursion still exists, the route remains quarantined. If the recursion has gone away, the route is
released from its quarantine.

The following steps are used to quarantine a route:

1. RIB detects when a particular problematic path is installed.
2. RIB sends a notification to the protocol that installed the path.
3. When the protocol receives the quarantine notification about the problem route, it marks the route as being
   "quarantined." If it is a BGP route, BGP does not advertise reachability for the route to its neighbors.
4. Periodically, RIB tests all its quarantined paths to see if they can now safely be installed (moved from
   quarantined to "Ok to use" state). A notification is sent to the protocol to indicate that the path is now safe
to use.

**Route and Label Consistency Checker**

The Route Consistency Checker and Label Consistency Checker (RCC/LCC) are command-line tools that
can be used to verify consistency between control plane and data plane route and label programming in IOS
XR software.

Routers in production networks may end up in a state where the forwarding information does not match the
control plane information. Possible causes of this include fabric or transport failures between the Route
Processor (RP) and the line cards (LCs), or issues with the Forwarding Information Base (FIB). RCC/LCC
can be used to identify and provide detailed information about resultant inconsistencies between the control
plane and data plane. This information can be used to further investigate and diagnose the cause of forwarding
problems and traffic loss.

RCC/LCC can be run in two modes. It can be triggered from EXEC mode as an on-demand, one-time scan
(On-demand Scan), or be configured to run at defined intervals in the background during normal router
operation (Background Scan). RCC compares the Routing Information Base (RIB) against the Forwarding
Information Base (FIB) while LCC compares the Label Switching Database (LSD) against the FIB. When an
inconsistency is detected, RCC/LCC output will identify the specific route or label and identify the type of
inconsistency detected as well as provide additional data that will assist with further troubleshooting.

RCC runs on the Route Processor. FIB checks for errors on the line card and forwards first the 20 error reports
to RCC. RCC receives error reports from all nodes, summarizes them (checks for exact match), and adds it
to two queues, soft or hard. Each queue has a limit of 1000 error reports and there is no prioritization in the
queue. RCC/LCC logs the same errors (exact match) from different nodes as one error. RCC/LCC compares
the errors based on prefix/label, version number, type of error, etc.

**On-demand Scan**

In On-demand Scan, user requests scan through the command line interface on a particular prefix in a particular
table or all the prefixes in the table. The scan is run immediately and the results are published right away.
LCC performs on-demand scan on the LSD, where as RCC performs it per VRF.
**Background Scan**

In Background Scan, user configures the scan that is then left to run in the background. The configuration consists of the time period for the periodic scan. This scan can be configured on either a single table or multiple tables. LCC performs background scan on the LSD, where as RCC performs it either for default or other VRFs.

**System-wide Route Prioritization for IOS XR Software**

System-wide route prioritization provides faster and more consistent Interior Gateway Protocol (IGP) convergence due to router or network events. This requires all routing components to prioritize important route updates from the routing clients (OSPF, ISIS, BGP, L2VPN), through the producer components (RIB, LSD) and the infrastructure (BCDL), to the FIB on all the line cards and route processors.

To support these requirements, the control plane needs to be able to prioritize its routes and download the critical routes first, followed by non-critical routes.

**System Flow**

The high-level description of system flow of prioritized route information is as follows:

- ISIS and OSPF pass a priority to RIB for the route
- RIB translates priority to BCDL queue and sends route to FIB
- If label is required for the route, RIB sends the route with priority to LDP
- LDP forwards the route with priority to LSD, using an LSD-supplied API
- LSD translates the priority to a specific internal LSD queue
- LSD maps the internal LSD queue to the corresponding BCDL queue and sends route to FIB

Applications like L2VPN, BGP Inter AS, and TE, which also require labels, have the following system flow:

- Application sends MPLS rewrites to LSD in the order of their internal priorities
- LSD translates the priority to a unique LSD internal queue
- LSD maps the internal LSD queue to a BCDL queue and sends rewrites to FIB.

**Note**

The priority queue for Route Consistency Checker (RCC) and Label Consistency Checker (LCC) will always be the lowest BCDL priority. Refer Route/Label Consistency Checker section for details on Route/Label Consistency Checker.

**How to Deploy and Monitor RIB**

To deploy and monitor RIB, you must understand the following concepts:
Verifying RIB Configuration Using the Routing Table

Perform this task to verify the RIB configuration to ensure that RIB is running on the RP and functioning properly by checking the routing table summary and details.

### SUMMARY STEPS

1. `show route [vrf {vrf-name | all}] [afi-all | ipv4 | ipv6] [unicast | multicast | safi-all] summary [detail] [standby]

2. `show route [vrf {vrf-name | all}] [afi-all | ipv4 | ipv6] [unicast | multicast | safi-all] [protocol [instance] | ip-address mask] [standby] [detail]

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Displays route summary information about the specified routing table.</td>
</tr>
<tr>
<td>`show route [vrf {vrf-name</td>
<td>all}] [afi-all</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router# show route summary</td>
</tr>
</tbody>
</table>

| **Step 2**        | Displays more detailed route information about the specified routing table. |
| `show route [vrf {vrf-name | all}] [afi-all | ipv4 | ipv6] [unicast | multicast | safi-all] [protocol [instance] | ip-address mask] [standby] [detail]` | • 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. |
| **Example:**      | RP/0/RSP0/CPU0:router# show route ipv4 unicast |

Verifying Networking and Routing Problems

Perform this task to verify the operation of routes between nodes.
SUMMARY STEPS

1. `show route [vrf {vrf-name | all}] [afi-all | ipv4 | ipv6] [unicast | multicast | safi-all] [protocol [instance] [ip-address mask] [standby] [detail]

2. `show route [vrf {vrf-name | all}] [afi-all | ipv4 | ipv6] [unicast | multicast | safi-all] backup [ip-address] [standby]

3. `show route [vrf {vrf-name | all}] [ipv4 | ipv6] [unicast | multicast | safi-all] best-local [ip-address] [standby]

4. `show route [vrf {vrf-name | all}] [afi-all | ipv4 | ipv6] [unicast | multicast | safi-all] local [interface] [standby]

5. `show route [vrf {vrf-name | all}] [afi-all | ipv4 | ipv6] [unicast | multicast | safi-all] longer-prefixes {ip-address mask | ip-address / prefix-length} [standby]

6. `show route [vrf {vrf-name | all}] [ipv4 | ipv6] [unicast | multicast | safi-all] best-local [ip-address] [standby]

7. `show route [vrf {vrf-name | all}] [ipv4 | ipv6] [unicast | multicast | safi-all] next-hop [ip-address] [standby]

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> show route [vrf {vrf-name</td>
<td>all}] [afi-all</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router# show route ipv4 unicast 192.168.1.11/8</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> show route [vrf {vrf-name</td>
<td>all}] [afi-all</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router# show route ipv4 unicast backup 192.168.1.11/8</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> show route [vrf {vrf-name</td>
<td>all}] [ipv4</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router# show route ipv4 unicast best-local 192.168.1.11/8</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> show route [vrf {vrf-name</td>
<td>all}] [afi-all</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router# show route ipv4 unicast connected</td>
<td></td>
</tr>
</tbody>
</table>
Disabling RIB Next-hop Dampening

Perform this task to disable RIB next-hop dampening.

SUMMARY STEPS

1. router rib
2. address-family { ipv4 | ipv6 } next-hop dampening disable
3. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 router rib</td>
<td>Enters RIB configuration mode.</td>
</tr>
</tbody>
</table>

Example:

```
RP/0/RSP0/CPU0:router# route rib
```
Configuring RCC and LCC

Enabling RCC and LCC On-demand Scan

Perform this task to trigger route consistency checker (RCC) and Label Consistency Checker (LCC) on-demand scan. The on-demand scan can be run on a particular address family (AFI), sub address family (SAFI), table and prefix, vrf, or all prefixes in the table.

**SUMMARY STEPS**

1. Use one of these commands.
   - `show rcc {ipv4 | ipv6} unicast [all] [prefix/mask] [vrf vrf-name]`
   - `show lcc {ipv4 | ipv6} unicast [all] [prefix/mask] [vrf vrf-name]`

2. Use one of these commands.
   - `clear rcc {ipv4 | ipv6} unicast [all] [prefix/mask] [vrf vrf-name] log`
   - `clear lcc {ipv4 | ipv6} unicast [all] [prefix/mask] [vrf vrf-name] log`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Runs on-demand Route Consistency Checker (RCC) or Label Consistency Checker (LCC).</td>
</tr>
<tr>
<td>Use one of these commands.</td>
<td></td>
</tr>
<tr>
<td>- `show rcc {ipv4</td>
<td>ipv6} unicast [all] [prefix/mask] [vrf vrf-name]`</td>
</tr>
<tr>
<td>- `show lcc {ipv4</td>
<td>ipv6} unicast [all] [prefix/mask] [vrf vrf-name]`</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router#show rcc ipv6 unicast 2001:DB8::/32 vrf vrf_1
```
### 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. **commit**
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>configure</strong></td>
</tr>
</tbody>
</table>

- **Example:**
  - `RP/0/RSP0/CPU0:router#clear rcc ipv6 unicast log`  
  - Or  
  - `RP/0/RSP0/CPU0:router#clear lcc ipv6 unicast log`  
  - Or  
  - `RP/0/RSP0/CPU0:router#show lcc ipv6 unicast log`
### Purpose

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

<table>
<thead>
<tr>
<th>Step 2</th>
<th>Use one of these commands:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• rcc {ipv4</td>
</tr>
<tr>
<td></td>
<td>• lcc {ipv4</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router(config)#rcc ipv6 unicast enable
RP/0/RSP0/CPU0:router(config)#rcc ipv6 unicast period 500
Or
RP/0/RSP0/CPU0:router(config)#lcc ipv6 unicast enable
RP/0/RSP0/CPU0:router(config)#lcc ipv6 unicast period 500
```

### Purpose

Displays statistics about background scans.

<table>
<thead>
<tr>
<th>Step 3</th>
<th>commit</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Step 4</th>
<th>Use one of these commands.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• show rcc {ipv4</td>
</tr>
<tr>
<td></td>
<td>• show lcc {ipv4</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RSP0/CPU0:router#show rcc ipv6 unicast statistics scan-id 120
Or
RP/0/RSP0/CPU0:router#show lcc ipv6 unicast statistics scan-id 120
```

---

**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

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

Next hop matches 0.0.0.0/0
Known via "static", distance 200, metric 0, candidate default path
Enabling RCC and LCC: Example

Enabling RCC and LCC Background Scan: Example

This example shows how to enable Route Consistency Checker (RCC) background scan with a period of 500 milliseconds between buffers in scans for IPv6 unicast tables:
```
rcc ipv6 unicast period 500
```

This example shows how to enable Label Consistency Checker (LCC) background scan with a period of 500 milliseconds between buffers in scans for IPv6 unicast tables:
```
lcc ipv6 unicast period 500
```

Enabling RCC and LCC On-demand Scan: Example

This example shows how to run Route Consistency Checker (RCC) on-demand scan for subnet 10.10.0.0/16 in vrf1:
```
show rcc ipv4 unicast 10.10.0.0/16 vrf vrf 1
```

This example shows how to run Label Consistency Checker (LCC) on-demand scan on all labels for IPv6 prefixes:
```
show lcc ipv6 unicast all
```

Where to Go Next

For additional information on the protocols that interact with RIB, you may want to see the following publications:

- Implementing MPLS Layer 3 VPNs in *Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide*
- Implementing BGP in *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide*
- Implementing EIGRP in *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide*
- Implementing IS-IS in *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide*
- Implementing OSPF in *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide*
- Implementing RIP in *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide*
- RIB Commands in *Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference*
## Additional References

### Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing Information Base commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td>RIB Commands on Cisco IOS XR Software in Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference</td>
</tr>
</tbody>
</table>

### Standards and RFCs

<table>
<thead>
<tr>
<th>Standard/RFC</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft-ietf-rtgwg-lf-conv-frmwk-00.txt</td>
<td>A Framework for Loop-free Convergence, by M. Shand and S. Bryant</td>
</tr>
<tr>
<td>No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>

### MIBs

<table>
<thead>
<tr>
<th>MIB</th>
<th>MIBs Link</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: <a href="http://www.cisco.com/go/mibs">http://www.cisco.com/go/mibs</a></td>
</tr>
</tbody>
</table>
### Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/support">http://www.cisco.com/support</a></td>
</tr>
</tbody>
</table>
CHAPTER 8

Implementing RIP

The Routing Information Protocol (RIP) is a classic distance vector Interior Gateway Protocol (IGP) designed to exchange information within an autonomous system (AS) of a small network.

This module describes the concepts and tasks to implement basic RIP routing. Cisco IOS XR software supports a standard implementation of RIP Version 2 (RIPv2) that supports backward compatibility with RIP Version 1 (RIPv1) as specified by RFC 2453.

For RIP configuration information related to the following features, see the Related Documents, on page 418 section of this module.

- Multiprotocol Label Switching (MPLS) Layer 3 Virtual Private Network (VPN)
- Site of Origin (SoO) Support

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 418 section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the Cisco ASR 9000 Series Aggregation Services Router Commands Master List.

Feature History for Implementing RIP

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.7.2</td>
<td>This feature was introduced.</td>
</tr>
<tr>
<td>Release 4.0.0</td>
<td>MD5 Authentication Using Keychain feature was added.</td>
</tr>
</tbody>
</table>

- Prerequisites for Implementing RIP, page 400
- Information About Implementing RIP, page 400
- How to Implement RIP, page 405
- Configuration Examples for Implementing RIP, page 415
- Additional References, page 418
Prerequisites for Implementing RIP

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Implementing RIP

RIP Functional Overview

RIP Version 1 (RIP v1) is a classful, distance-vector protocol that is considered the easiest routing protocol to implement. Unlike OSPF, RIP broadcasts User Datagram Protocol (UDP) data packets to exchange routing information in internetworks that are flat rather than hierarchical. Network complexity and network management time is reduced. However, as a classful routing protocol, RIP v1 allows only contiguous blocks of hosts, subnets or networks to be represented by a single route, severely limiting its usefulness.

RIP v2 allows more information carried in RIP update packets, such as support for:

- Route summarization
- Classless interdomain routing (CIDR)
- Variable-length subnet masks (VLSMs)
- Autonomous systems and the use of redistribution
- Multicast address 224.0.0.9 for RIP advertisements

The metric that RIP uses to rate the value of different routes is hop count. The hop count is the number of routers that can be traversed in a route. A directly connected network has a metric of zero; an unreachable network has a metric of 16. This small range of metrics makes RIP an unsuitable routing protocol for large networks.

Routing information updates are advertised every 30 seconds by default, and new updates discovered from neighbor routers are stored in a routing table.

Only RIP Version 2 (RIP v2), as specified in RFC 2453, is supported on Cisco IOS XR software and, by default, the software only sends and receives RIP v2 packets. However, you can configure the software to send, or receive, or both, only Version 1 packets or only Version 2 packets or both version type packets per interface.

Here are some good reasons to use RIP:

- Compatible with diverse network devices
- Best for small networks, because there is very little overhead, in terms of bandwidth used, configuration, and management time
- Support for legacy host systems

Because of RIP’s ease of use, it is implemented in networks worldwide.
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.

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 407 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 4: Default Administrative Distances of Routing Protocols

<table>
<thead>
<tr>
<th>Routing Protocols</th>
<th>Administrative Distance Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected interface</td>
<td>0</td>
</tr>
<tr>
<td>Static route out an interface</td>
<td>0</td>
</tr>
<tr>
<td>Static route to next hop</td>
<td>1</td>
</tr>
<tr>
<td>EIGRP Summary Route</td>
<td>5</td>
</tr>
<tr>
<td>Routing Protocols</td>
<td>Administrative Distance Value</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>External BGP</td>
<td>20</td>
</tr>
<tr>
<td>Internal EIGRP</td>
<td>90</td>
</tr>
<tr>
<td>OSPF</td>
<td>110</td>
</tr>
<tr>
<td>IS-IS</td>
<td>115</td>
</tr>
<tr>
<td>RIP version 1 and 2</td>
<td>120</td>
</tr>
<tr>
<td>External EIGRP</td>
<td>170</td>
</tr>
<tr>
<td>Internal BGP</td>
<td>200</td>
</tr>
<tr>
<td>Unknown</td>
<td>255</td>
</tr>
</tbody>
</table>

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 Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide
- Keychain Management Commands module in Cisco ASR 9000 Series Aggregation Services Router System Security Command Reference

Note

The keychain by itself has no relevance; therefore, it must be used by an application that needs to communicate by using the keys (for authentication) with its peers. The keychain provides a secure mechanism to handle the keys and rollover based on the lifetime. The Cisco IOS XR keychain infrastructure takes care of the hit-less rollover of the secret keys in the keychain.

Once you have configured a keychain in the IOS XR keychain database and if the same has been configured on a particular RIP interface, it will be used for authenticating all incoming and outgoing RIP traffic on that interface. Unless an authentication keychain is configured on a RIP interface (on the default VRF or a non-default VRF), all RIP traffic will be assumed to be authentic and authentication mechanisms for in-bound RIP traffic and out-bound RIP traffic will 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.

<table>
<thead>
<tr>
<th>If...</th>
<th>Then...</th>
</tr>
</thead>
<tbody>
<tr>
<td>The keychain configured on the RIP interface does not exist in the keychain database...</td>
<td>The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.</td>
</tr>
<tr>
<td>The keychain is not configured with a MD5 cryptographic algorithm...</td>
<td>The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.</td>
</tr>
<tr>
<td>The Address Family Identifier of the first (and only the first) entry in the message is not 0xFFFF, then authentication is not in use...</td>
<td>The packet will be dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.</td>
</tr>
</tbody>
</table>
Then... If... The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.
The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.

<table>
<thead>
<tr>
<th>If...</th>
<th>Then...</th>
</tr>
</thead>
<tbody>
<tr>
<td>The MD5 digest in the ‘Authentication Data’ is found to be invalid...</td>
<td>The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.</td>
</tr>
<tr>
<td>Else, the packet is forwarded for the rest of the processing.</td>
<td></td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>If...</th>
<th>Then...</th>
</tr>
</thead>
<tbody>
<tr>
<td>The keychain configured on the RIP interface exists in the keychain database ...</td>
<td>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.</td>
</tr>
<tr>
<td>The keychain is configured with a MD5 cryptographic algorithm...</td>
<td>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.</td>
</tr>
<tr>
<td>Else, RIP packets fail authentication check.</td>
<td></td>
</tr>
</tbody>
</table>

**How to Implement RIP**

This section contains instructions for the following tasks:

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

**Enabling RIP**

This task enables RIP routing and establishes a RIP routing process.

**Before You Begin**

Although you can configure RIP before you configure an IP address, no RIP routing occurs until at least one IP address is configured.
### SUMMARY STEPS

1. `configure`
2. `router rip`
3. `neighbor ip-address`
4. `broadcast-for-v2`
5. `interface type interface-path-id`
6. `receive version { 1 | 2 | 1 2 }`
7. `send version { 1 | 2 | 1 2 }`
8. `commit`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> <code>configure</code></td>
<td>Configures a RIP routing process.</td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>router rip</code></td>
<td>Configures a RIP routing process.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config)# router rip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>neighbor ip-address</code></td>
<td>(Optional) Defines a neighboring router with which to exchange RIP protocol information.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-rip)# neighbor 172.160.1.2</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> <code>broadcast-for-v2</code></td>
<td>(Optional) Configures RIP to send only Version 2 packets to the broadcast IP address rather than the RIP v2 multicast address (224.0.0.9). This command can be applied at the interface or global configuration level.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-rip)# broadcast-for-v2</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> <code>interface type interface-path-id</code></td>
<td>(Optional) Defines the interfaces on which the RIP routing protocol runs.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-rip)# interface GigabitEthernet 0/1/0/0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> `receive version { 1</td>
<td>2</td>
</tr>
</tbody>
</table>
| **Example:**<br>RP/0/RSP0/CPU0:router(config-rip-if)# receive version 1 2 | • Only RIP v1  
• Only RIP v2  
• Both RIP v1 and RIP v2 |
| **Step 7** `send version { 1 | 2 | 1 2 }` | (Optional) Configures an interface to send packets that are: |
Customizing RIP

This task describes how to customize RIP for network timing and the acceptance of route entries.

SUMMARY STEPS

1. configure
2. router rip
3. auto-summary
4. timers basic update invalid holddown flush
5. output-delay delay
6. nsf
7. interface type interface-path-id
8. metric-zero-accept
9. split-horizon disable
10. poison-reverse
11. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Configures a RIP routing process.</td>
</tr>
</tbody>
</table>
| **Step 2** router rip | (Optional) Enables automatic route summarization of subnet routes into network-level routes.
• By default, auto-summary is disabled. |
| Example: 
RP/0/RSP0/CPU0:router(config)# router rip | |
| **Step 3** auto-summary | |
| Example: 
RP/0/RSP0/CPU0:router(config-rip)# auto-summary | |
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 4</strong> timers basic update invalid holddown flush</td>
<td>Note: If you have disconnected subnets, use the <code>no</code> keyword to disable automatic route summarization and permit software to send subnet and host routing information across classful network boundaries.</td>
</tr>
<tr>
<td>Example:</td>
<td>(Optional) Adjusts RIP network timers.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rip)# timers basic 5 15 15 30</td>
<td>Note: To view the current and default timer values, view output from the <code>show rip</code> command.</td>
</tr>
<tr>
<td><strong>Step 5</strong> output-delay delay</td>
<td>(Optional) Changes the interpacket delay for the RIP updates sent.</td>
</tr>
<tr>
<td>Example:</td>
<td>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.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rip)# output-delay 10</td>
<td>(Optional) Configures NSF on RIP routes after a RIP process shutdown or restart.</td>
</tr>
<tr>
<td><strong>Step 6</strong> nsf</td>
<td>(Optional) Defines the interfaces on which the RIP routing protocol runs.</td>
</tr>
<tr>
<td>Example:</td>
<td>(Optional) Configures NSF on RIP routes after a RIP process shutdown or restart.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rip)# nsf</td>
<td>(Optional) Defines the interfaces on which the RIP routing protocol runs.</td>
</tr>
<tr>
<td><strong>Step 7</strong> interface type interface-path-id</td>
<td>(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).</td>
</tr>
<tr>
<td>Example:</td>
<td>(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).</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rip)# interface GigabitEthernet 0/1/0/0</td>
<td>(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).</td>
</tr>
<tr>
<td><strong>Step 8</strong> metric-zero-accept</td>
<td>(Optional) Enables the split horizon mechanism.</td>
</tr>
<tr>
<td>Example:</td>
<td>(Optional) Enables the split horizon mechanism.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rip-if)# metric-zero-accept</td>
<td>• By default, split horizon is enabled.</td>
</tr>
<tr>
<td><strong>Step 9</strong> split-horizon disable</td>
<td>• In general, we do not recommend changing the state of the default for the <code>split-horizon</code> command, unless you are certain that your application requires a change to properly advertise routes. If split horizon is disabled on a serial interface (and that interface is attached to a packet-switched network), you must disable split horizon for all networking devices in any relevant multicast groups on that network.</td>
</tr>
<tr>
<td>Example:</td>
<td>(Optional) Enables the split horizon mechanism.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rip-if)# split-horizon disable</td>
<td>(Optional) Enables the split horizon mechanism.</td>
</tr>
</tbody>
</table>
Control Routing Information

This task describes how to control or prevent routing update exchange and propagation.

Some reasons to control or prevent routing updates are:

- To slow or stop the update traffic on a WAN link—If you do not control update traffic on an on-demand WAN link, the link remains up constantly. By default, RIP routing updates occur every 30 seconds.

- To prevent routing loops—If you have redundant paths or are redistributing routes into another routing domain, you may want to filter the propagation of one of the paths.

- To filter network received in updates—If you do not want other routers from learning a particular device’s interpretation of one or more routes, you can suppress that information.

- To prevent other routers from processing routes dynamically—If you do not want to process routing updates entering the interface, you can suppress that information.

- To preserve bandwidth—You can ensure maximum bandwidth availability for data traffic by reducing unnecessary routing update traffic.

**SUMMARY STEPS**

1. configure
2. router rip
3. neighbor ip-address
4. interface type interface-path-id
5. passive-interface
6. exit
7. interface type interface-path-id
8. route-policy { in | out }
9. commit

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 10 poison-reverse</td>
<td>Enables poison reverse processing of RIP router updates.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-rip-if)# poison-reverse</td>
<td></td>
</tr>
</tbody>
</table>

Step 11 commit
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router rip</td>
<td>Configures a RIP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config)# router rip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> neighbor <code>ip-address</code></td>
<td>(Optional) Defines a neighboring router with which to exchange RIP protocol information.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-rip)# neighbor 172.160.1.2</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> interface <code>type interface-path-id</code></td>
<td>(Optional) Defines the interfaces on which the RIP routing protocol runs.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-rip)# interface GigabitEthernet 0/1/0/0</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> passive-interface</td>
<td>(Optional) Suppresses the sending of RIP updates on an interface, but not to explicitly configured neighbors.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-rip-if)# passive-interface</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> exit</td>
<td>(Optional) Returns the router to the next higher configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-rip-if)# exit</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> interface <code>type interface-path-id</code></td>
<td>(Optional) Defines the interfaces on which the RIP routing protocol runs.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-rip)# interface GigabitEthernet 0/2/0/0</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> route-policy `{ in</td>
<td>out }`</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-rip-if)# route-policy out</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>
Creating a Route Policy for RIP

This task defines a route policy and shows how to attach it to an instance of a RIP process. Route policies can be used to:

- Control routes sent and received
- Control which routes are redistributed
- Control origination of the default route

A route policy definition consists of the `route-policy` command and `name` argument followed by a sequence of optional policy statements, and then closes with the `end-policy` command.

A route policy is not useful until it is applied to routes of a routing protocol.

**SUMMARY STEPS**

1. configure
2. route-policy `name`
3. set rip-metric `number`
4. end-policy
5. commit
6. configure
7. router rip
8. route-policy `route-policy-name { in | out }
9. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Defines a route policy and enters route-policy configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> route-policy <code>name</code></td>
<td></td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# route-policy IN-IPv4</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> set rip-metric <code>number</code></td>
<td>(Optional) Sets the RIP metric attribute.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-rpl)# set rip metric 42</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> end-policy</td>
<td>End the definition of a route policy and exits route-policy configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-rpl)# end-policy</td>
<td></td>
</tr>
</tbody>
</table>
### Configuring RIP Authentication Keychain

#### Configuring RIP Authentication Keychain for IPv4 Interface on a Non-default VRF

Perform this task to configure a RIP authentication keychain for IPv4 interface on a non-default VRF.

**Before You Begin**

All keychains need to be configured in Cisco IOS XR keychain database using configuration commands described in Implementing Keychain Management module of Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide before they can be applied to a RIP interface/VRF.

The authentication keychain `keychain-name` and mode `md5` configurations will accept the name of a keychain that has not been configured yet in the IOS XR keychain database or a keychain that has been configured in IOS XR keychain database without MD5 cryptographic algorithm. However, in both these cases, all incoming packets on the interface will be dropped and outgoing packets will be sent without authentication data.

---

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 5</td>
<td>commit</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td>router rip</td>
<td>Configures a RIP routing process.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# router rip</td>
<td></td>
</tr>
<tr>
<td>Step 8</td>
<td>route-policy route-policy-name { in</td>
<td>out }</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-rip)# route-policy rpl in</td>
<td></td>
</tr>
<tr>
<td>Step 9</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>
SUMMARY STEPS

1. configure
2. router rip
3. vrf vrf_name
4. interface type interface-path-id
5. Use one of these commands:
   - authentication keychain keychain-name mode md5
   - authentication keychain keychain-name mode text
6. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Configures a RIP routing process.</td>
</tr>
<tr>
<td><strong>Step 2</strong> router rip</td>
<td>Configures a RIP routing process.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)#router rip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> vrf vrf_name</td>
<td>Configures a non-default VRF</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-rip)#vrf vrf_rip_auth</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> interface type interface-path-id</td>
<td>Defines the interface on which the RIP routing protocol runs.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-rip-vrf)#interface POS 0/6/0/0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> Use one of these commands:</td>
<td>Configures an authentication keychain mode for RIP.</td>
</tr>
<tr>
<td>· authentication keychain keychain-name mode md5</td>
<td>· md5—Keyed message digest (md5) authentication mode</td>
</tr>
<tr>
<td>· authentication keychain keychain-name mode text</td>
<td>· text—Clear text authentication mode</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-rip-if)#authentication keychain key1 mode md5</td>
<td></td>
</tr>
<tr>
<td>Or</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rip-if)#authentication keychain key1 mode text</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>
Configuring RIP Authentication Keychain for IPv4 Interface on Default VRF

Perform this task to configure a RIP authentication keychain for IPv4 interface (on the default VRF).

**Before You Begin**

All keychains need to be configured in Cisco IOS XR keychain database using configuration commands described in *Implementing Keychain Management* module of *Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide* before they can be applied to a RIP interface/VRF.

The **authentication keychain** `keychain-name` and **mode md5** configurations will accept the name of a keychain that has not been configured yet in the IOS XR keychain database or a keychain that has been configured in IOS XR keychain database without MD5 cryptographic algorithm. However, in both these cases, all incoming packets on the interface will be dropped and outgoing packets will be sent without authentication data.

**SUMMARY STEPS**

1. `configure`
2. `router rip`
3. `interface type interface-path-id`
4. Use one of these commands:
   - `authentication keychain keychain-name mode md5`
   - `authentication keychain keychain-name mode text`
5. `commit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>router rip</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config)#router rip</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Configures a RIP routing process.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><code>interface type interface-path-id</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-rip)#interface POS 0/6/0/0</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Defines the interface on which the RIP routing protocol runs.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Use one of these commands:</td>
</tr>
<tr>
<td></td>
<td>• <code>authentication keychain keychain-name mode md5</code></td>
</tr>
<tr>
<td></td>
<td>• <code>authentication keychain keychain-name mode text</code></td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Configures an authentication keychain mode for RIP.</td>
</tr>
<tr>
<td></td>
<td>• md5——Keyed message digest (md5) authentication mode</td>
</tr>
</tbody>
</table>
### 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.

```plaintext
interface GigabitEthernet0/6/0/0
  ipv4 address 172.16.0.1 255.255.255.0
!
interface GigabitEthernet0/6/0/2
  ipv4 address 172.16.2.12 255.255.255.0
!
router rip
  interface GigabitEthernet0/6/0/0
  !
  interface GigabitEthernet0/6/0/2
  !
```

#### Configuring RIP on the Provider Edge: Example

The following example shows how to configure basic RIP on the PE with two VPN routing and forwarding (VRF) instances.

```plaintext
router rip
  interface GigabitEthernet0/6/0/0
  !
  vrf vpn0
    interface GigabitEthernet0/6/0/2
    !
  vrf vpn1
    interface GigabitEthernet0/6/0/3
    !
```
Adjusting RIP Timers for each VRF Instance: Example

The following example shows how to adjust RIP timers for each VPN routing and forwarding (VRF) instance. For VRF instance vpn0, the timers basic command sets updates to be broadcast every 10 seconds. If a router is not heard from in 30 seconds, the route is declared unusable. Further information is suppressed for an additional 30 seconds. At the end of the flush period (45 seconds), the route is flushed from the routing table. For VRF instance vpn1, timers are adjusted differently: 20, 60, 60, and 70 seconds. The output-delay command changes the interpacket delay for RIP updates to 10 milliseconds on vpn1. The default is that interpacket delay is turned off.

```
router rip
    interface GigabitEthernet0/6/0/0
    !
    vrf vpn0
        interface GigabitEthernet0/6/0/2
        !
            timers basic 10 30 30 45
        !
        vrf vpn1
            interface GigabitEthernet0/6/0/3
            !
                timers basic 20 60 60 70
                output-delay 10
            !
```

Configuring Redistribution for RIP: Example

The following example shows how to redistribute Border Gateway Protocol (BGP) and static routes into RIP. The RIP metric used for redistributed routes is determined by the route policy. If a route policy is not configured or the route policy does not set RIP metric, the metric is determined based on the redistributed protocol. For VPNv4 routes redistributed by BGP, the RIP metric set at the remote PE router is used, if valid.

In all other cases (BGP, IS-IS, OSPF, EIGRP, connected, static), the metric set by the default-metric command is used. If a valid metric cannot be determined, then redistribution does not happen.

```
route-policy ripred
    set rip-metric 5
end-policy
!

router rip
    vrf vpn0
        interface GigabitEthernet0/6/0/2
        !
            redistribute connected
default-metric 3
        !
    vrf vpn1
        interface GigabitEthernet0/6/0/3
        !
            redistribute bgp 100 route-policy ripred
default-metric 3
        !
```

Configuring Route Policies for RIP: Example

The following example shows how to configure inbound and outbound route policies that are used to control which route updates are received by a RIP interface or sent out from a RIP interface.

```plaintext
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.

```plaintext
router rip
  interface GigabitEthernet0/6/0/0
    passive-interface
  !
  interface GigabitEthernet0/6/0/2
  !
  neighbor 172.17.0.1
  neighbor 172.18.0.5
  !
```
Controlling RIP Routes: Example

The following example shows how to use the `distance` command to install RIP routes in the Routing Information Base (RIB). The `maximum-paths` command controls the number of maximum paths allowed per RIP route.

```
router rip
  interface GigabitEthernet0/6/0/0
  route-policy polin in

  distance 110
  maximum-paths 8

```

Configuring RIP Authentication Keychain: Example

This example shows how to apply an authentication keychain on a RIP default VRF interface:

```
router rip
  interface POS0/6/0/0
  authentication keychain key1 mode md5

```

This example shows how to apply an authentication keychain on a RIP non-default interface:

```
router rip
  vrf rip_keychain_vrf
  interface POS0/6/0/0
  authentication keychain key1 mode md5

```

Additional References

The following sections provide references related to implementing RIP.

### Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td><em>Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference</em></td>
</tr>
<tr>
<td>MPLS VPN support for RIP feature information</td>
<td>Implementing MPLS Traffic Engineering on Cisco ASR 9000 Series Router module in the Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide</td>
</tr>
</tbody>
</table>
### Related Topic

<table>
<thead>
<tr>
<th>Site of Origin (SoO) support for RIP feature information</th>
<th>Implementing MPLS Traffic Engineering on Cisco ASR 9000 Series Router module in the Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XR getting started documentation</td>
<td>Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide</td>
</tr>
<tr>
<td>Information about user groups and task IDs</td>
<td>Configuring AAA Services on Cisco ASR 9000 Series Router module in the Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide</td>
</tr>
</tbody>
</table>

### Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>

### MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>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: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></td>
</tr>
</tbody>
</table>

### RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 2453</td>
<td>RIP Version 2</td>
</tr>
</tbody>
</table>
## Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a></td>
</tr>
</tbody>
</table>
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.

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 498 section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the Cisco ASR 9000 Series Aggregation Services Router Commands Master List.

### Feature History for Implementing Routing Policy

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.7.2</td>
<td>This feature was introduced.</td>
</tr>
<tr>
<td>Release 3.9.0</td>
<td>Parameterization was supported at all attach points.</td>
</tr>
</tbody>
</table>
| Release 4.2.0 | The following features were added:  
  • Hierarchical Conditions  
  • Apply Condition Policies |
Prerequisites for Implementing Routing Policy

The following are prerequisites for implementing Routing Policy on Cisco IOS XR Software:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

- Border Gateway Protocol (BGP), integrated Intermediate System-to-Intermediate System (IS-IS), or Open Shortest Path First (OSPF) must be configured in your network.

Restrictions for Implementing Routing Policy

These restrictions apply when working with Routing Policy Language implementation on Cisco IOS XR software:

- An individual policy definition of up to 1000 statements are supported. The total number of statements within a policy can be extended to 4000 statements using hierarchical policy constructs. However, this limit is restricted with the use of `apply` statements.

- When a policy that is attached directly or indirectly to an attach point needs to be modified, a single `commit` operation cannot be performed when:
  
  - Removing a set or policy referred by another policy that is attached to any attach point directly or indirectly.
  
  - Modifying the policy to remove the reference to the same set or policy that is getting removed.

The `commit` must be performed in two steps:
1. Modify the policy to remove the reference to the policy or set and then commit.
2. Remove the policy or set and commit.

Information About Implementing Routing Policy

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

Routing Policy Language

This section contains the following information:

Routing Policy Language Overview

RPL was developed to support large-scale routing configurations. RPL has several fundamental capabilities that differ from those present in configurations oriented to traditional route maps, access lists, and prefix lists. The first of these capabilities is the ability to build policies in a modular form. Common blocks of policy can be defined and maintained independently. These common blocks of policy can then be applied from other blocks of policy to build complete policies. This capability reduces the amount of configuration information that needs to be maintained. In addition, these common blocks of policy can be parameterized. This parameterization allows for policies that share the same structure but differ in the specific values that are set or matched against to be maintained as independent blocks of policy. For example, three policies that are identical in every way except for the local preference value they set can be represented as one common parameterized policy that takes the varying local preference value as a parameter to the policy.

The policy language introduces the notion of sets. Sets are containers of similar data that can be used in route attribute matching and setting operations. Four set types exist: prefix-sets, community-sets, as-path-sets, and extcommunity-sets. These sets hold groupings of IPv4 or IPv6 prefixes, community values, AS path regular expressions, and extended community values, respectively. Sets are simply containers of data. Most sets also have an inline variant. An inline set allows for small enumerations of values to be used directly in a policy rather than having to refer to a named set. Prefix lists, community lists, and AS path lists must be maintained even when only one or two items are in the list. An inline set in RPL allows the user to place small sets of values directly in the policy body without having to refer to a named set.

Decision making, such as accept and deny, is explicitly controlled by the policy definitions themselves. RPL combines matching operators, which may use set data, with the traditional Boolean logic operators AND, OR, and NOT into complex conditional expressions. All matching operations return a true or false result. The execution of these conditional expressions and their associated actions can then be controlled by using simple if then, elseif, and else structures, which allow the evaluation paths through the policy to be fully specified by the user.

Routing Policy Language Structure

This section describes the basic structure of RPL.
Names

The policy language provides two kinds of persistent, namable objects: sets and policies. Definition of these objects is bracketed by beginning and ending command lines. For example, to define a policy named test, the configuration syntax would look similar to the following:

```
route-policy test
  [. . . policy statements . . .]
end-policy
```

Legal names for policy objects can be any sequence of the upper- and lowercase alphabetic characters; the numerals 0 to 9; and the punctuation characters period, hyphen, and underscore. A name must begin with a letter or numeral.

Sets

In this context, the term set is used in its mathematical sense to mean an unordered collection of unique elements. The policy language provides sets as a container for groups of values for matching purposes. Sets are used in conditional expressions. The elements of the set are separated by commas. Null (empty) sets are allowed.

In the following example:

```
prefix-set backup-routes
  # currently no backup routes are defined
end-set
```

a condition such as:

```
if destination in backup-routes then
```

evaluates as FALSE for every route, because there is no match-condition in the prefix set that it satisfies.

Five kinds of sets exist: as-path-set, on page 425, community-set, on page 426, extcommunity-set, on page 427, prefix-set, on page 430, and rd-set, on page 431. 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:

```plaintext
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 matching operations are:

- dfa-regex: DFA style regular expression
- ios-regex: Traditional IOS style regular expression
- length: Length of BGP AS-path
- neighbor-is: BGP AS-path neighbor is
- originates-from: BGP AS-path originates-from
- passes-through: BGP AS-path passes-through
- unique-length: Length of BGP AS-path ignoring duplicates

**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:

```plaintext
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.
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'.

**Note**

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 435 for more information.

RPL provides symbolic names for the standard well-known community values: internet is 0:0, no-export is 65535:65281, no-advertise is 65535:65282, and local-as is 65535is-empty:65283.

RPL also provides a facility for using wildcards in community specifications. A wildcard is specified by inserting an asterisk (*) in place of one of the 16-bit portions of the community specification; the wildcard indicates that any value for that portion of the community matches. Thus, the following policy matches all communities in which the autonomous system part of the community is 123:

```
community-set cset3
123:*
end-set
```
A community set can either be empty, or contain one or more community values. When used with an empty community set, the **is-empty** operator will evaluate to **TRUE** and the **matches-any** and **matches-every** operators will evaluate to **FALSE**.

**extcommunity-set**

An extended community-set is analogous to a community-set except that it contains extended community values instead of regular community values. It also supports named forms and inline forms. There are three types of extended community sets: cost, soo, and rt.

As with community sets, the inline form supports parameterization within parameterized policies. Either portion of the extended community value can be parameterized.

Wildcards (*) and regular expressions are allowed for extended community set elements.

Every extended community-set must contain at least one extended community value. Empty extended community-sets are invalid and rejected.

The following are syntactic examples:

**Named Form for Extcommunity-set Cost**

A cost set is an extcommunity set used to store cost EIGRP Cost Community type extended community type communities.

```
  extcommunity-set cost a_cost_set
  IGP:1:10
  end-set
```

These options are supported under extended community set Cost:

```
RP/0/RSP0/CPU0:router(config) extcommunity-set cost cost_set
RP/0/RSP0/CPU0:router(config-ext)#
  #-remark   Remark beginning with '#'
  <0-255>    decimal number
  abort      Discard RPL definition and return to top level config
  end-set    End of set definition
  exit       Exit from this submode
  igp:       Cost Community with IGP as point of insertion
  pre-bestpath: Cost Community with Pre-Bestpath as point of insertion
  show       Show partial RPL configuration
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#-remark</td>
<td>Remark beginning with '#'</td>
</tr>
<tr>
<td>&lt;0-255&gt;</td>
<td>decimal number</td>
</tr>
<tr>
<td>abort</td>
<td>Discard RPL definition and return to top level config</td>
</tr>
<tr>
<td>end-set</td>
<td>End of set definition</td>
</tr>
<tr>
<td>exit</td>
<td>Exit from this submode</td>
</tr>
<tr>
<td>igp:</td>
<td>Cost Community with IGP as point of insertion</td>
</tr>
<tr>
<td>pre-bestpath:</td>
<td>Cost Community with Pre-Bestpath as point of insertion</td>
</tr>
<tr>
<td>show</td>
<td>Show partial RPL configuration</td>
</tr>
</tbody>
</table>
Named Form for Extcommunity-set RT

An rt set is an extcommunity set used to store BGP Route Target (RT) extended community type communities:

```
extcommunity-set rt a_rt_set
    1.2.3.4:666
    1234:666,
    1.2.3.4:777,
    4567:777
end-set
```

Inline Set Form for Extcommunity-set RT

```
(1.2.3.4:666, 1234:666, 1.2.3.4:777, 4567:777)
($ipadrr:666, 1234:$tag, 1.2.3.4:777, $tag2:777)
```

These options are supported under extended community set RT:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#-remark</td>
<td>Remark beginning with '#'</td>
</tr>
<tr>
<td>*</td>
<td>Wildcard (any community or part thereof)</td>
</tr>
<tr>
<td>&lt;1-4294967295&gt;</td>
<td>32-bit decimal number</td>
</tr>
<tr>
<td>&lt;1-65535&gt;</td>
<td>16-bit decimal number</td>
</tr>
<tr>
<td>A.B.C.D/M:N</td>
<td>Extended community - IPv4 prefix format</td>
</tr>
<tr>
<td>A.B.C.D:N</td>
<td>Extended community - IPv4 format</td>
</tr>
<tr>
<td>ASN:N</td>
<td>Extended community - ASPLAIN format</td>
</tr>
<tr>
<td>X.Y:N</td>
<td>Extended community - ASDOT format</td>
</tr>
<tr>
<td>abort</td>
<td>Discard RPL definition and return to top level config</td>
</tr>
<tr>
<td>dfa-regex</td>
<td>DFA style regular expression</td>
</tr>
<tr>
<td>end-set</td>
<td>End of set definition</td>
</tr>
<tr>
<td>exit</td>
<td>Exit from this subnode</td>
</tr>
<tr>
<td>ios-regex</td>
<td>Traditional IOS style regular expression</td>
</tr>
<tr>
<td>show</td>
<td>Show partial RPL configuration</td>
</tr>
</tbody>
</table>
Named Form for Extcommunity-set Soo

A soo set is an extcommunity set used to store BGP Site-of-Origin (SoO) extended community type communities:

```plaintext
extcommunity-set soo a_soo_set
  1.1.1:100,
  100:200
end-set
```

These options are supported under extended community set Soo:

```plaintext
RP/0/RSP0/CPU0:router(config)#extcommunity-set soo soo_set
RP/0/RSP0/CPU0:router(config-ext)#?
#-remark Remark beginning with '#'
* Wildcard (any community or part thereof)
<1-4294967295> 32-bit decimal number
<1-65535> 16-bit decimal number
A.B.C.D/M:N Extended community - IPv4 prefix format
A.B.C.D:N Extended community - IPv4 format
ASN:N Extended community - ASPLAIN format
X.Y:N Extended community - ASDOT format
abort Discard RPL definition and return to top level config
dfa-regex DFA style regular expression
dfa-regex DFA style regular expression
end-set End of set definition
exit Exit from this submode
ios-regex Traditional IOS style regular expression
show Show partial RPL configuration
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#-remark</td>
<td>Remark beginning with '#'</td>
</tr>
<tr>
<td>*</td>
<td>Wildcard (any community or part thereof)</td>
</tr>
<tr>
<td>&lt;1-4294967295&gt;</td>
<td>32-bit decimal number</td>
</tr>
<tr>
<td>&lt;1-65535&gt;</td>
<td>16-bit decimal number</td>
</tr>
<tr>
<td>A.B.C.D/M:N</td>
<td>Extended community - IPv4 prefix format</td>
</tr>
<tr>
<td>A.B.C.D:N</td>
<td>Extended community - IPv4 format</td>
</tr>
<tr>
<td>ASN:N</td>
<td>Extended community - ASPLAIN format</td>
</tr>
<tr>
<td>X.Y:N</td>
<td>Extended community - ASDOT format</td>
</tr>
<tr>
<td>abort</td>
<td>Discard RPL definition and return to top level config</td>
</tr>
<tr>
<td>dfa-regex</td>
<td>DFA style regular expression</td>
</tr>
<tr>
<td>end-set</td>
<td>End of set definition</td>
</tr>
<tr>
<td>exit</td>
<td>Exit from this submode</td>
</tr>
<tr>
<td>ios-regex</td>
<td>Traditional IOS style regular expression</td>
</tr>
<tr>
<td>show</td>
<td>Show partial RPL configuration</td>
</tr>
</tbody>
</table>
prefix-set

A prefix-set holds IPv4 or IPv6 prefix match specifications, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. The address is a standard dotted-decimal IPv4 or colon-separated hexadecimal IPv6 address. The mask length, if present, is a nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6) following the address and separated from it by a slash. The optional minimum matching length follows the address and optional mask length and is expressed as the keyword `ge` (mnemonic for greater than or equal to), followed by a nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6). The optional maximum matching length follows the rest and is expressed by the keyword `le` (mnemonic for less than or equal to), followed by yet another nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6). A syntactic shortcut for specifying an exact length for prefixes to match is the `eq` keyword (mnemonic for equal to).

If a prefix match specification has no mask length, then the default mask length is 32 for IPv4 and 128 for IPv6. The default minimum matching length is the mask length. If a minimum matching length is specified, then the default maximum matching length is 32 for IPv4 and 128 for IPv6. Otherwise, if neither minimum nor maximum is specified, the default maximum is the mask length.

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 10.0.4.240/24. The fifth element matches prefixes in the range 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.[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.

**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:*`—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, 10.0.0.2:255.255.0.00.
- `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, 10002:255.255.0.00.
- `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 490 for detailed information on how to implement routing policy language.

**Pass Policy**

The following example shows how the policy accepts all presented routes without modifying the routes.

```plaintext
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.

```plaintext
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.

```plaintext
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.

```conf
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.

```conf
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.

```conf
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:

```plaintext
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.

```plaintext
route-policy origin-10
if as-path originates-from '10.5' then
apply param-example(10.5)
else
pass
endif
end-policy
```

```plaintext
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 435, 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 "$smymed" and "$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.

```
routerr 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
    glbpblahype 'ebgp'
    glbtltag '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 `gbpblahype` and `glbtltag`, is defined for a nonparameterized policy.

```
route-policy globalparam
    if path-type is $gbpblahype then
        set tag $gbtltltag
    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:

\[
\text{med eq 10 and not destination in (10.1.3.0/24) or community matches-any ([10..25]:35)}
\]

if fully parenthesized to display the order of evaluation, would look like this:

\[
(\text{med eq 10 and (not destination in (10.1.3.0/24))) or community matches-any ([10..25]:35)}
\]

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:

\[
\text{not med eq 10 and destination in (10.1.3.0/24) or community matches-any ([10..25]:35)}
\]

then the expression, fully parenthesized, would look like the following:

\[
((\text{not med eq 10}) \text{ and destination in (10.1.3.0/24)}) \text{ or community matches-any ([10..25]:35)}
\]

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:

\[
\begin{align*}
\text{set med 9} \\
\text{set med 10} \\
\text{set med 11} \\
\text{set med 12}
\end{align*}
\]

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:

\[
\begin{align*}
\text{set med 8} \\
\text{if community matches-any cs1 then} \\
\text{set local-preference 122} \\
\text{if community matches-any cs2 then} \\
\text{set med 12} \\
\text{endif} \\
\text{endif}
\end{align*}
\]

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:

\[
\text{route-policy community-add} \\
\text{set community (10:23)}
\]
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:

```
if med 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:

```plaintext
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:

```plaintext
route-policy four-equivalent
set weight 100
set med 200
set community (2:666) additive
pass
end-policy
```

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.

```plaintext
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.

```plaintext
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.

```plaintext
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.

```plaintext
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:

```plaintext
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:

```plaintext
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:

```plaintext
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:

```plaintext
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.

**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:

```plaintext
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:

```plaintext
not next-hop in (10.0.2.2)
```

Any Boolean condition enclosed in parentheses is itself a Boolean condition:

```plaintext
(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 bgp 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 bgp 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 *Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference* 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
```

### Aggregation

The aggregation attach point generates an aggregate route to be advertised based on the conditional presence of subcomponents of that aggregate. Policies attached at this attach point are also able to set any of the valid BGP attributes on the aggregated routes. For example, the policy could set a community value or a MED on the aggregate that is generated. The specified aggregate is generated if any routes evaluated by the named policy pass the policy. More specifics of the aggregate are filtered using the `suppress-route` keyword. Any actions taken to set attributes in the route affect attributes on the aggregate.

In the policy language, the configuration is controlled by which routes pass the policy. The suppress map was used to selectively filter or suppress specific components of the aggregate when the summary-only flag is not set. In other words, when the aggregate and more specific components are being sent, some of the more specific components can be filtered using a suppress map. In the policy language, this is controlled by selecting the route and setting the suppress flag. The attribute-map allowed the user to set specific attributes on the aggregated route. In the policy language, setting attributes on the aggregated route is controlled by normal action operations.

In the following example, the aggregate address 10.0.0.0/8 is generated if there are any component routes in the range 10.0.0.0/8 ge 8 le 25 except for 10.2.0.0/24. Because summary-only is not set, all components of the aggregate are advertised. However, the specific component 10.1.0.0 are suppressed.

```
route-policy sample
  if destination in (10.0.0.0/8 ge 8 le 25) then
    set community (10:33)
  endif
  if destination in (10.2.0.0/24) then
    drop
  endif
  if destination in (10.1.0.0/24) then
    suppress-route
  endif
end-policy
router bgp 2
  address-family ipv4
    aggregate-address 10.0.0.0/8 route-policy sample
```

The effect of aggregation policy on the attributes of the aggregate is cumulative. Every time an aggregation policy matches a more specific route, the set operations in the policy may modify the aggregate. The aggregate in the following example has a MED value that varies according to the number of more specific routes that comprise the aggregate.

```
route-policy bumping-aggregation
  set med +5
end-policy
```

If there are three matching more specific routes, the MED of the aggregate is the default plus 15; if there are seventeen more specific routes, the MED of the aggregate is the default plus 85.
The order that the aggregation policy is applied to prefix paths is deterministic but unspecified. That is, a given set of routes always appears in the same order, but there is no way to predict the order.

A drop in aggregation policy does not prevent generation of an aggregate, but it does prevent the current more specific route from contributing to the aggregate. If another more specific route gives the route a pass, the aggregate is generated. Only one more specific pass is required to generate an aggregate.

**Dampening**

The dampening attach point controls the default route-dampening behavior within BGP. Unless overridden by a more specific policy on the associate peer, all routes in BGP apply the associated policy to set their dampening attributes.

The following policy sets dampening values for BGP IPv4 unicast routes. Those routes that are more specific than a /25 take longer to recover after they have been dampened than routes that are less specific than /25.

```plaintext
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.

```plaintext
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 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 (BRII) 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
```
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 allows routes from other sources to be advertised by BGP. The policy attached at this point is able to set any of the valid BGP attributes on the routes that are being redistributed. Likewise, selection operators allow a user to control what route sources are being redistributed and which routes from those sources.

The following example shows how to redistribute all routes from OSPF instance 12 into BGP. If OSPF were carrying a default route, it is dropped. Routes carrying a tag of 10 have their local preference set to 300 and the community value of 2:666 and no-advertise attached. All other routes have their local preference set to 200 and a community value of 2:100 set.

```
route-policy sample_redistribute
  if destination in (0.0.0.0/0) then
    drop
  endif
  if tag eq 10 then
    set local-preference 300
    set community (2:666, no-advertise)
  else
    set local-preference 200
    set community (2:100)
  endif
end-policy

router bgp 2
  address-family ipv4 unicast
    redistribute ospf 12 route-policy sample_redistribute
```
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
```

```
Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x
OL-26048-03
451
Implementing Routing Policy  Attach Points
```
MET ORG AS
  origin:incomplete  neighbor as:3  metric:333
  aspath:2 3

Table Policy

The table policy attach point allows the user to configure traffic-index values on routes as they are installed into the global routing table. This attach point supports the BGP policy accounting feature. BGP policy accounting uses the traffic indexes that are set on the BGP routes to track various counters. This way, router operators can select different sets of BGP route attributes using the matching operations and then set different traffic indexes for each different class of route they are interested in tracking.

The following example shows how to set the traffic index to 10 in IPv4 unicast routes that originated from autonomous system 10.33. Likewise, any IPv4 unicast routes that originated from autonomous system 11.60 have their traffic index set to 11 when they are installed into the FIB. These traffic indexes are then used to count traffic being forwarded on these routes inline cards by enabling the BGP policy accounting counters on the interfaces of interest.

```
route-policy sample-table
  if as-path originates-from '10.33' then
    set traffic-index 10
  elseif as-path originates-from '11.60' then
    set traffic-index 11
  endif
end-policy
router bgp 2
  address-family ipv4 unicast
    table-policy sample-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 bgpvrfs_import
  if extcommunity rt matches-any (10:91) then
    set next-hop 172.16.0.1
  elseif extcommunity rt matches-every (11:92) then
    set next-hop 172.16.0.2
  elseif extcommunity soo matches-any (10:111111, 10:111222) then
    pass
  endif
```
end-policy

vrf vrf_import
  address-family ipv4 unicast
  import route-policy bgpvrf_import

Note
'Set' is a valid operator for the 'med' attribute at the bgp import attach point.

Export

The export attach point provides control over the export of routes from a particular VRF to a global VPN IPv4 table.

For Layer 3 VPN networks, export route targets are added to the VPN IPv4 routes when VRF IPv4 routes are converted into VPN IPv4 routes and advertised through the MP-iBGP to other PE routers (or flow from one VRF to another within a PE router).

A set of export route targets is configured with the VRF without RPL configuration; however, to set route targets conditionally, you can configure a VRF export policy.

The following example shows some match and set operations supported for the export route policy. If a route matches 172.16.1.0/24 then the route target extended community is set to 10:101, and the weight is set to 211. If the route does not match 172.16.1.0/24 but the origin of the route is egp, then the local preference is set to 212 and the route target extended community is set to 10:101. If the route does not match those specified criteria, then the route target extended community 10:111222 is added to the route. In addition, RT 10:111222 is added to the route that matches any of the previous conditions as well.

route-policy bgpvrf_export
  if destination in (172.16.1.0/24) then
    set extcommunity rt (10:101)
    set weight 211
  elseif origin is egp then
    set local-preference 212
    set extcommunity rt (10:101)
  endif
  set extcommunity rt (10:111222) additive
end-policy

vrf vrf-export
  address-family ipv4 unicast
  export route-policy bgpvrf-export

Note
'Set' is a valid operator for the 'med' attribute at the bgp export attach point.
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
      .
    .
```

Allocate-Label

The allocate-label attach point provides increased control based on various attribute match operations. This attach point is typically used in inter-AS option C to decide whether the label should be allocated or not when sending updates to the neighbor for the IPv4 labeled unicast address family. The attribute setting actions supported are for pass and drop.

The following example shows how to configure a route policy that passes the prefix 0.0.0.0 with prefix length 0. Label allocation happens only if prefix 0.0.0.0 exists.

```
route-policy label_policy
  if destination in (0.0.0.0/0) then
    pass
  endif
end-policy

router bgp 2
  vrf vrf1
    rd auto
    address-family ipv4 unicast
      allocate-label route-policy label-policy
        .
        .
```

Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x
**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
end-policy

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
        nxthop route-policy nxthp_policy_A
```

Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x
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.

```plaintext
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.

```plaintext
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 5: BGP Attributes and Operators

<table>
<thead>
<tr>
<th>Attach Point</th>
<th>Attribute</th>
<th>Match</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>additional-paths</td>
<td>path-selection</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>community</td>
<td>matches-every</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-empty</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td></td>
</tr>
<tr>
<td>aggregation</td>
<td>as-path</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>originates-from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>as-path-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>as-path-unique-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>community</td>
<td>is-empty</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td>delete in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete not in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete all</td>
</tr>
<tr>
<td></td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>extcommunity</td>
<td></td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>cost</td>
<td></td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td>local-preference</td>
<td>is, ge, le, eq</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>med</td>
<td>is, eg, ge, le</td>
<td>setset +set -</td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>origin</td>
<td>is</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>source</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>suppress-route</td>
<td>—</td>
<td>suppress-route</td>
</tr>
<tr>
<td></td>
<td>weight</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>allocate-label</td>
<td>as-path</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>originates-from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td>as-path-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>as-path-unique-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>community</td>
<td></td>
<td>is-empty</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td></td>
</tr>
<tr>
<td>destination</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>label</td>
<td></td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td>local-preference</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>med</td>
<td></td>
<td>is, eg, ge, le</td>
<td></td>
</tr>
<tr>
<td>next-hop</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>origin</td>
<td></td>
<td>is</td>
<td></td>
</tr>
<tr>
<td>source</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>clear-policy</td>
<td>as-path</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>originates-from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td>as-path-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>as-path-unique-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------</td>
<td>--------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>dampening</td>
<td>as-path</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>originates-from passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td>as-path-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>as-path-unique-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>community</td>
<td></td>
<td>is-empty matches-any matches-every</td>
<td></td>
</tr>
<tr>
<td>dampening</td>
<td></td>
<td>—/</td>
<td>set dampening</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(to set values that control the dampening, see Dampening, on page 448)</td>
</tr>
<tr>
<td>destination</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>local-preference</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>med</td>
<td></td>
<td>is, eg, ge, le</td>
<td></td>
</tr>
<tr>
<td>next-hop</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>origin</td>
<td></td>
<td>is</td>
<td></td>
</tr>
<tr>
<td>source</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>debug</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>default originate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rib-has-route</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td><strong>Attach Point</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>Match</strong></td>
<td><strong>Set</strong></td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>export (VRF)</td>
<td>as-path</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>originates-from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td>as-path-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td>as-path-unique-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td>community</td>
<td></td>
<td>is-empty</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td>additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>set</td>
<td>delete in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete not in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete all</td>
<td></td>
</tr>
<tr>
<td>destination</td>
<td></td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>extcommunity rt</td>
<td></td>
<td>is-empty</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td>additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-within</td>
<td>delete-in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete-not-in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete-all</td>
<td></td>
</tr>
<tr>
<td>extcommunity soo</td>
<td></td>
<td>is-empty</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-within</td>
<td></td>
</tr>
<tr>
<td>local-preference</td>
<td></td>
<td>is, ge, le, eq</td>
<td>set</td>
</tr>
<tr>
<td>med</td>
<td></td>
<td>is, eg, ge, le</td>
<td>set</td>
</tr>
<tr>
<td>next-hop</td>
<td></td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>origin</td>
<td></td>
<td>is</td>
<td>—</td>
</tr>
<tr>
<td>source</td>
<td></td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>weight</td>
<td></td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------</td>
<td>------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>import (VRF)</td>
<td>as-path</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is originates-from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>passes-through unique-length</td>
<td></td>
</tr>
<tr>
<td>as-path-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>as-path-unique-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>community</td>
<td></td>
<td>is-empty</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td></td>
</tr>
<tr>
<td>destination</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>extcommunity rt</td>
<td></td>
<td>is-empty</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-within</td>
<td></td>
</tr>
<tr>
<td>extcommunity soo</td>
<td></td>
<td>is-empty</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-within</td>
<td></td>
</tr>
<tr>
<td>local-preference</td>
<td></td>
<td>is, ge, le, eq</td>
<td>set</td>
</tr>
<tr>
<td>med</td>
<td></td>
<td>is, eg, ge, le</td>
<td>set</td>
</tr>
<tr>
<td>next-hop</td>
<td></td>
<td>in</td>
<td>set peer address set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>set destination vrf</td>
<td></td>
</tr>
<tr>
<td>origin</td>
<td></td>
<td>is</td>
<td></td>
</tr>
<tr>
<td>source</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------</td>
<td>------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>neighbor-in</td>
<td>as-path</td>
<td>in</td>
<td>prepend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local</td>
<td>prepend-most-recent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>length</td>
<td>replace</td>
</tr>
<tr>
<td>as-path-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td>as-path-unique-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td>community</td>
<td>community</td>
<td>is-empty</td>
<td>set</td>
</tr>
<tr>
<td>with 'peeras'</td>
<td></td>
<td>matches-any</td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td>delete-in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete-not-in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete-all</td>
</tr>
<tr>
<td>destination</td>
<td>in</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td>extcommunity cost</td>
<td>—</td>
<td>—</td>
<td>set additive</td>
</tr>
<tr>
<td>extcommunity rt</td>
<td></td>
<td>is-empty</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td>delete-in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-within</td>
<td>delete-not-in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete-all</td>
</tr>
<tr>
<td>extcommunity soo</td>
<td></td>
<td>is-empty</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-within</td>
<td>—</td>
</tr>
<tr>
<td>local-preference</td>
<td>is, ge, le, eq</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td>med</td>
<td>is, eg, ge, le</td>
<td></td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set +</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set -</td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------</td>
<td>------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>next-hop</td>
<td>in</td>
<td>set</td>
<td>set peer address</td>
</tr>
<tr>
<td>origin</td>
<td>is</td>
<td>set</td>
<td></td>
</tr>
<tr>
<td>source</td>
<td>in</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weight</td>
<td>—</td>
<td>set</td>
<td></td>
</tr>
<tr>
<td>neighbor-out</td>
<td>as-path</td>
<td>in</td>
<td>is-local length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>neighbor-is originates-from passes-through unique-length</td>
</tr>
<tr>
<td>as-path-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>as-path-unique-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>community with 'peeras'</td>
<td>is-empty</td>
<td>set</td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td>matches-any</td>
<td>delete-in</td>
<td>delete-not-in</td>
</tr>
<tr>
<td></td>
<td>matches-every</td>
<td>delete-all</td>
<td></td>
</tr>
<tr>
<td>destination</td>
<td>in</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>extcommunity cost</td>
<td>—</td>
<td>set</td>
<td>set additive</td>
</tr>
<tr>
<td>extcommunity rt</td>
<td>is-empty</td>
<td>set</td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td>matches-any</td>
<td>delete-in</td>
<td>delete-not-in</td>
</tr>
<tr>
<td></td>
<td>matches-every</td>
<td>delete-all</td>
<td></td>
</tr>
<tr>
<td>extcommunity soo</td>
<td></td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>-------</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-empty matches-any matches-every matches-within</td>
<td></td>
</tr>
<tr>
<td>local-preference</td>
<td></td>
<td>is, ge, le, eq</td>
<td>set</td>
</tr>
<tr>
<td>med</td>
<td></td>
<td>is, eg, ge, le</td>
<td>set + set - set max-unreachable set igr-cost</td>
</tr>
<tr>
<td>next-hop</td>
<td></td>
<td>in</td>
<td>set set self</td>
</tr>
<tr>
<td>origin</td>
<td></td>
<td>is</td>
<td>set</td>
</tr>
<tr>
<td>path-type</td>
<td></td>
<td>is</td>
<td>—</td>
</tr>
<tr>
<td>rd</td>
<td></td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>source</td>
<td></td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>unsuppress-route</td>
<td></td>
<td>—</td>
<td>unsuppress-route</td>
</tr>
<tr>
<td>vpn-distinguisher</td>
<td></td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td>neighbor-orf</td>
<td>orf-prefix</td>
<td>in</td>
<td>n/a</td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>network</td>
<td>as-path</td>
<td>—</td>
<td>prepend</td>
</tr>
<tr>
<td>community</td>
<td>—</td>
<td>set</td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete-in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete-not-in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete-all</td>
<td></td>
</tr>
<tr>
<td>destination</td>
<td>in</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>extcommunity cost</td>
<td>—</td>
<td>set</td>
<td>set additive</td>
</tr>
<tr>
<td>mpls-label</td>
<td>route-has-label</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>local-preference</td>
<td>—</td>
<td>set</td>
<td></td>
</tr>
<tr>
<td>med</td>
<td>—</td>
<td>set</td>
<td>set+</td>
</tr>
<tr>
<td></td>
<td></td>
<td>set-</td>
<td></td>
</tr>
<tr>
<td>next-hop</td>
<td>in</td>
<td>set</td>
<td></td>
</tr>
<tr>
<td>origin</td>
<td>—</td>
<td>set</td>
<td></td>
</tr>
<tr>
<td>route-type</td>
<td>is</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td>is, ge, le, eq</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>weight</td>
<td>—</td>
<td>set</td>
<td></td>
</tr>
<tr>
<td>next-hop</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>protocol</td>
<td>is,in</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>source</td>
<td>in</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>redistribute</td>
<td>as-path</td>
<td>—</td>
<td>prepend</td>
</tr>
<tr>
<td></td>
<td>community</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete not in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete all</td>
</tr>
<tr>
<td></td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>extcommunity cost</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td>local-preference</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>med</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set-</td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>origin</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>mpls-label</td>
<td>route-has-label</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>route-type</td>
<td>is</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>is, eq, ge, le</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>weight</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td>retain-rt</td>
<td>extcommunity rt</td>
<td>is-empty</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-within</td>
<td></td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>-------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>show</td>
<td>as-path</td>
<td>in is-local length</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is originates-from passes-through unique-length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>as-path-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>as-path-unique-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>community</td>
<td>is-empty matches-any matches-every</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>extcommunity rt</td>
<td>is-empty matches-any matches-every matches-within</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>extcommunity soo</td>
<td>is-empty matches-any matches-every matches-within</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>med</td>
<td>is, eg, ge, le</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>origin</td>
<td>is</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>source</td>
<td>in</td>
<td>—</td>
</tr>
</tbody>
</table>
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. This table summarizes which operations are valid and where they are valid.

**Table 6: Restricted BGP Operations by Attach Point**

<table>
<thead>
<tr>
<th>Command</th>
<th>import</th>
<th>export</th>
<th>aggregation</th>
<th>redistribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>prepend as-path</td>
<td>EBGP only</td>
<td>EBGP only</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>most-recent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Attach Point</th>
<th>Attribute</th>
<th>Match</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>table-policy</td>
<td>as-path</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>originates-from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>as-path-unique-length</td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td></td>
<td>community</td>
<td>is-empty</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td></td>
</tr>
<tr>
<td></td>
<td>local-preference</td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td></td>
<td>destination</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>med</td>
<td>is, eg, ge, le</td>
<td></td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>origin</td>
<td>is</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rib-metric</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>source</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>traffic-index</td>
<td>—</td>
<td>set</td>
</tr>
</tbody>
</table>
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.

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.

<table>
<thead>
<tr>
<th>Command</th>
<th>import</th>
<th>export</th>
<th>aggregation</th>
<th>redistribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace as-path</td>
<td>eBGP only</td>
<td>eBGP only</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>set med igp-cost</td>
<td>forbidden</td>
<td>eBGP only</td>
<td>forbidden</td>
<td>forbidden</td>
</tr>
<tr>
<td>set weight</td>
<td>n/a</td>
<td>forbidden</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>suppress</td>
<td>forbidden</td>
<td>forbidden</td>
<td>n/a</td>
<td>forbidden</td>
</tr>
</tbody>
</table>

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

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.
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:

```plaintext
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.

```plaintext
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 10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24, it is accepted. If the prefix matches any of 10.106.3.0/24, 10.106.7.0/24, 10.106.13.0/24, it is dropped.

```
route-policy OSPF-area-in
  if destination in (10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24) then
    drop
  endif
  if destination in (10.106.3.0/24, 10.106.7.0/24, 10.106.13.0/24) 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 10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24, it is announced. If the prefix matches any of 10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24, it is dropped and not announced.

```
route-policy OSPF-area-out
  if destination in (10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24) then
    drop
  endif
  if destination in (10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24) 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 7: OSPF Attributes and Operators**

<table>
<thead>
<tr>
<th>Attach Point</th>
<th>Attribute</th>
<th>Match</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>distribute-list-in-area</td>
<td>destination</td>
<td>in</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>rib-metric</td>
<td>in</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>eq, ge, is, le</td>
<td>n/a</td>
</tr>
<tr>
<td>distribute-list-in-instance</td>
<td>destination</td>
<td>in</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>rib-metric</td>
<td>in</td>
<td>n/a</td>
</tr>
<tr>
<td>distribute-list-in-interface</td>
<td>destination</td>
<td>in</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>rib-metric</td>
<td>in</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>eq, ge, is, le</td>
<td>n/a</td>
</tr>
<tr>
<td>default-information originate</td>
<td>ospf-metric</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>metric-type</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>rib-has-route</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>redistribute</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>metric-type</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>ospf-metric</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>mpls-label</td>
<td>route-has-label</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>rib-metric</td>
<td>is, le, ge, eq</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>route-type</td>
<td>is</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>is, eq, ge, le</td>
<td>set</td>
</tr>
</tbody>
</table>
### Distribute-list in

The distribute-list in attach point within OSPF allows use of route policies to filter OSPF prefixes. The distribute-list in route-policy can be configured at OSPF instance, area, and interface levels. The route-policy used in the distribute-list in command supports match statements, "destination" and "rib-metric". The "set" commands are not supported in the route-policy.

These are examples of valid route-policies for "distribute-list in":

```plaintext
route-policy DEST
  if destination in (10.10.10.10/32) then
    drop
  else
    pass
  endif
end-policy

route-policy METRIC
  if rib-metric ge 10 and rib-metric le 19 then
    drop
  else
    pass
  endif
end-policy

prefix-set R-PFX
  10.10.10.30
end-set

route-policy R-SET
  if destination in R-PFX and rib-metric le 20 then
    pass
  else
    drop
  endif
end-policy
```

### OSPFv3 Policy Attach Points

This section describes each of the OSPFv3 policy attach points and provides a summary of the OSPFv3 attributes and operators.
Default-Information Originate

The default-information originate attach point allows the user to conditionally inject the default route 0::/0 into the OSPFv3 link-state database, which is done by evaluating the attached policy. If any routes in the local RIB pass the policy, then the default route is inserted into the link-state database.

The following example shows how to generate a default route if any of the routes that match 2001::/96 are present in the RIB:

```plaintext
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.

```plaintext
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.
### 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.

```plaintext
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.

```plaintext
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.

```plaintext
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 9: IS-IS Attributes and Operators

<table>
<thead>
<tr>
<th>Attach Point</th>
<th>Attribute</th>
<th>Match</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>redistribution</td>
<td>tag</td>
<td>is, eq, ge, le</td>
<td>set</td>
</tr>
<tr>
<td>route-type</td>
<td>is</td>
<td>Note The following route-type cannot be matched: ospf-nssa-type-1 and ospf-nssa-type-2</td>
<td>—</td>
</tr>
<tr>
<td>destination</td>
<td>in</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>next-hop</td>
<td>in</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>mpls-label</td>
<td>route-has-label</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>level</td>
<td>—</td>
<td>set</td>
<td>—</td>
</tr>
<tr>
<td>isis-metric</td>
<td>—</td>
<td>set</td>
<td>—</td>
</tr>
<tr>
<td>metric-type</td>
<td>—</td>
<td>set</td>
<td>—</td>
</tr>
<tr>
<td>default-information originate</td>
<td>rib-has-route</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>level</td>
<td>—</td>
<td>set</td>
<td>—</td>
</tr>
<tr>
<td>isis-metric</td>
<td>—</td>
<td>set</td>
<td>—</td>
</tr>
<tr>
<td>tag</td>
<td>—</td>
<td>set</td>
<td>—</td>
</tr>
<tr>
<td>inter-area-propagate</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
</tbody>
</table>

**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:

```plaintext
route-policy eigrp-cd-policy-in
    if destination in (10.0.0.0/8 ge 8 le 25) then
        pass
    endif
end-policy
```

The following examples show a policy that sets the conditional default flag for all routes that match 10.0.0.0/16:

```plaintext
route-policy eigrp-cd-policy-out
    if destination in (10.0.0.0/16) then
        pass
    endif
end-policy
```

**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 10.0.0.0/16:

```plaintext
route-policy eigrp-cd-policy-out
    if destination in (10.0.0.0/16) then
        pass
    endif
end-policy
```

**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:

```plaintext
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 10: EIGRP Attributes and Operators*

<table>
<thead>
<tr>
<th>Attach Point</th>
<th>Attribute</th>
<th>Match</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>default-accept-in</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>default-accept-out</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>if-policy-in</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>eigrp-metric</td>
<td>—</td>
<td>add, set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>is, eq, ge, le</td>
<td>set</td>
</tr>
<tr>
<td>if-policy-out</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>protocol</td>
<td>is, in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>eigrp-metric</td>
<td>—</td>
<td>add, set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>is, eq, ge, le</td>
<td>set</td>
</tr>
<tr>
<td>policy-in</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>eigrp-metric</td>
<td>—</td>
<td>add, set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>is, eq, ge, le</td>
<td>set</td>
</tr>
<tr>
<td>policy-out</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>protocol</td>
<td>is, in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>eigrp-metric</td>
<td>—</td>
<td>add, set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>is, eq, ge, le</td>
<td>set</td>
</tr>
</tbody>
</table>
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
```

RIP Attributes and Operators

This table summarizes the RIP attributes and operators per attach points.
### Table 11: RIP Attributes and Operators

<table>
<thead>
<tr>
<th>Attach Point</th>
<th>Attribute</th>
<th>Match</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>default-information</td>
<td>next-hop</td>
<td>na</td>
<td>set</td>
</tr>
<tr>
<td>originate</td>
<td>rip-metric</td>
<td>na</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>rip-tag</td>
<td>na</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>rib-has-route</td>
<td>in</td>
<td>na</td>
</tr>
<tr>
<td>global-inbound</td>
<td>destination</td>
<td>in</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>rip-metric</td>
<td>na</td>
<td>add</td>
</tr>
<tr>
<td>global-outbound</td>
<td>destination</td>
<td>in</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>protocol</td>
<td>is, in</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>rip-metric</td>
<td>na</td>
<td>add</td>
</tr>
<tr>
<td>interface-inbound</td>
<td>destination</td>
<td>in</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>rip-metric</td>
<td>na</td>
<td>add</td>
</tr>
<tr>
<td>interface-outbound</td>
<td>destination</td>
<td>in</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>protocol</td>
<td>is, in</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>rip-metric</td>
<td>na</td>
<td>add</td>
</tr>
<tr>
<td>redistribute</td>
<td>destination</td>
<td>in</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>rip-metric</td>
<td>na</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>rip-tag</td>
<td>na</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>mpls-label</td>
<td>route-has-label</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>route-type</td>
<td>is</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>is, eq, ge, le</td>
<td>set</td>
</tr>
</tbody>
</table>
PIM Policy Attach Points

This section describes the PIM policy `rpf-topology` attach point and provides a summary of the PIM attributes and operators.

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 reentering 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.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

Nonattached Policy Modification

As long as a given policy is not attached at an attach point, the policy is allowed to refer to nonexistent sets and policies. Configurations can be built that reference sets or policy blocks that are not yet defined, and then later those undefined policies and sets can be filled in. This method of building configurations gives much greater flexibility in policy definition. Every piece of policy you want to reference while defining a policy need not exist in the configuration. Thus, you can define a policy `sample1` that references a policy `sample2` using an `apply` statement even if the policy `sample2` does not exist. Similarly, you can enter a policy statement that refers to a nonexistent set.

However, the existence of all referenced policies and sets is enforced when a policy is attached. Thus, if a user attempts to attach the policy `sample1` with the reference to an undefined policy `sample2` at an inbound BGP policy using the statement `neighbor 1.2.3.4 address-family ipv4 unicast policy sample1 in`, the configuration attempt is rejected because the policy `sample2` does not exist.

Editing Routing Policy Configuration Elements

RPL is based on statements rather than on lines. That is, within the begin-end pair that brackets policy statements from the CLI, a new line is merely a separator, the same as a space character.

The CLI provides the means to enter and delete route policy statements. RPL provides a means to edit the contents of the policy between the begin-end brackets, using a text editor. The following text editors are available on Cisco IOS XR software for editing RPL policies:

- Nano (default)
- Emacs
- Vim
**Editing Routing Policy Configuration Elements Using the Nano Editor**

To edit the contents of a routing policy using the Nano editor, use the following CLI command in EXEC mode:

```plaintext
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/](http://www.nano-editor.org/).

Not all Nano editor features are supported on Cisco IOS XR software.

**Editing Routing Policy Configuration Elements Using the Emacs Editor**

To edit the contents of a routing policy using the Emacs editor, use the following CLI command in EXEC mode:

```plaintext
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/RSP0/CPU0:router# edit route-policy policy_A
微 EMACS 3.8b () -- rpl_edit.139281 --
  if destination in (2001::/8) then
    drop
  endif
end-policy
!
微 EMACS 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.
```
If there are parse errors, you are asked whether editing should continue:

```
RP/0/RSP0/CPU0:router#edit route-policy policy_B
-- MicroEMACS 3.8b () -- rpl_edit:141738
route-policy policy_B
set metric-type type_1
if destination in (2001::/8) then
drop
endif
end-policy
!
-- MicroEMACS 3.8b () -- rpl_edit:141738 --
Parsing.
105 bytes parsed in 1 sec (103)bytes/sec
% Syntax/Authorization errors in one or more commands.!! CONFIGURATION
FAILED DUE TO SYNTAX/AUTHORIZATION ERRORS
set metric-type type_1
if destination in (2001::/8) then
drop
endif
end-policy
!
Continue editing? [no]:
```

If you answer yes, the editor continues on the text buffer from where you left off. If you answer no, the running configuration is not changed and the editing session is ended.

**Editing Routing Policy Configuration Elements Using the Vim Editor**

Editing elements of a routing policy with Vim (Vi IMproved) is similar to editing them with Emacs except for some feature differences such as the keystrokes to save and quit. To write to a current file and exit, use the `:wq` or `:x` or `ZZ` keystrokes. To quit and confirm, use the `:q` keystrokes. To quit and discard changes, use the `:q!` keystrokes.

You can reference detailed online documentation for Vim at this URL: http://www.vim.org/

**Editing Routing Policy Configuration Elements Using CLI**

The CLI allows you to enter and delete route policy statements. You can complete a policy configuration block by entering applicable commands such as `end-policy` or `end-set`. Alternatively, the CLI interpreter allows you to use the `exit` command to complete a policy configuration block. The `abort` command is used to discard the current policy configuration and return to global configuration mode.

**Editing Routing Policy Language set elements Using XML**

RPL supports editing set elements using XML. Entries can be appended, prepended, or deleted to an existing set without replacing it through XML.
Hierarchical Policy Conditions

The Hierarchical Policy Conditions feature enables the ability to specify a route policy within the "if" statement of another route policy. This ability enables route-policies to be applied for configurations that are based on hierarchical policies.

With the Hierarchical Policy Conditions feature, Cisco IOS XR RPL supports Apply Condition policies that can be used with various types of Boolean operators along with various other matching statements.

Apply Condition Policies

Apply Condition policies, which Cisco IOS XR RPL supports, allow usage of a route-policy within an "if" statement of another route-policy.

Consider route-policy configurations Parent, Child A, and Child B:

```
route-policy Child A
  if destination in (10.10.0.0/16) then
    set local-pref 111
  endif
end-policy
!
route-policy Child B
  if as-path originates-from '222' then
    set community (333:222) additive
  endif
end-policy
!
route-policy Parent
  if apply Child A and apply Child B then
    set community (333:333) additive
  else
    set community (333:444) additive
  endif
end-policy
!
```

In the above scenarios, whenever the policy Parent is executed, the decision of the "if" condition in that is selected based on the result of policies Child A and Child B. The policy Parent is equivalent to policy merged as given below:

```
route-policy merged
  if destination in (10.10.0.0/16) and as-path originates-from '222' then
    set local-pref 111
    set community (333:222, 333:333) additive
  elseif destination in (10.10.0.0/16) then /*Only Policy Child A is pass */
    set local-pref 111
    set community (333:444) additive /*From else block */
  elseif as-path originates-from '222' then /*Only Policy Child B is pass */
    set community (333:222, 333:444) additive /*From else block */
  else
    set community (333:444) additive /*From else block */
  endif
end-policy
```

Apply Conditions can be used with parameters and are supported on all attach points and on all clients. Hierarchical Apply Conditions can be used without any constraints on a cascaded level.

Existing route policy semantics can be expanded to include this Apply Condition:

```
Route-policy policy_name
  If apply policyA and apply policyB then
    Set med 100
```

```
Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide, Release 4.2.x
OL-26048-03
487
```
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.

<table>
<thead>
<tr>
<th>Route-policies with simple hierarchical policies</th>
<th>Possible done statement execution sequence</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>pass</td>
<td>pass</td>
<td>Marks the prefix as &quot;acceptable&quot; and continues with execution of continue_list statements.</td>
</tr>
<tr>
<td></td>
<td>Continue_list</td>
<td></td>
</tr>
<tr>
<td>drop</td>
<td>Stmts_list</td>
<td>Rejects the route immediately on hitting the drop statement and stops policy execution.</td>
</tr>
<tr>
<td></td>
<td>drop</td>
<td></td>
</tr>
<tr>
<td>done</td>
<td>Stmts_list</td>
<td>Accepts the route immediately on hitting the done statement and stops policy execution.</td>
</tr>
<tr>
<td></td>
<td>done</td>
<td></td>
</tr>
<tr>
<td>pass followed by done</td>
<td>pass</td>
<td>Exits immediately at the done statement with &quot;accept route&quot;.</td>
</tr>
<tr>
<td></td>
<td>Statement_list</td>
<td></td>
</tr>
<tr>
<td></td>
<td>done</td>
<td></td>
</tr>
<tr>
<td>drop followed by done</td>
<td>drop</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Statement_list</td>
<td></td>
</tr>
<tr>
<td></td>
<td>done</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Hierarchical policy conditions</th>
<th>Possible done statement execution sequence</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>pass</td>
<td>pass</td>
<td>Marks the return value as &quot;true&quot; and continues execution within the same policy condition. If there is no statement after &quot;pass&quot;, returns &quot;true&quot;.</td>
</tr>
<tr>
<td>pass followed by done</td>
<td>pass or set action statement Stmt_list done</td>
<td>Marks the return value as &quot;true&quot; and continues execution till the done statement. Returns &quot;true&quot; to the apply policy condition to take &quot;true-path&quot;.</td>
</tr>
<tr>
<td>done</td>
<td>Stmt_list without pass or set operation DONE</td>
<td>Returns &quot;false&quot;. Condition takes &quot;false-path&quot;.</td>
</tr>
<tr>
<td>drop</td>
<td>Stmt_list drop Stmt_list</td>
<td>The prefix is dropped or rejected.</td>
</tr>
</tbody>
</table>

**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:

```plaintext
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
```
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 Using Wildcards For Routing Policy Sets: Example, on page 496.

How to Implement Routing Policy

This section contains the following procedures:

Defining a Route Policy

This task explains how to define a route policy.

- If you want to modify an existing routing policy using the command-line interface (CLI), you must redefine the policy by completing this task.
- Modifying the RPL scale configuration may take a long time.
- BGP may crash either due to large scale RPL configuration changes, or during consecutive RPL changes. To avoid BGP crash, wait until there are no messages in the BGP In/Out queue before committing further changes.

SUMMARY STEPS

1. configure
2. route-policy  name  [ parameter1 , parameter2 ,..., parameterN ]
3. end-policy
4. commit
DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters route-policy configuration mode.</td>
</tr>
<tr>
<td>Step 2 route-policy name [parameter1, parameter2,..., parameterN]</td>
<td>Enters route-policy configuration mode.</td>
</tr>
</tbody>
</table>

Example:
```
RP/0/RSP0/CPU0:router(config)# route-policy sample1
```

<table>
<thead>
<tr>
<th>Step 3 end-policy</th>
<th>Ends the definition of a route policy and exits route-policy configuration mode.</th>
</tr>
</thead>
</table>

Example:
```
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
```

| Step 4 commit | |

Attaching a Routing Policy to a BGP Neighbor

This task explains how to attach a routing policy to a BGP neighbor.

Before You Begin

A routing policy must be preconfigured and well defined prior to it being applied at an attach point. If a policy is not predefined, an error message is generated stating that the policy is not defined.

SUMMARY STEPS

1. configure
2. router bgp as-number
3. neighbor ip-address
4. address-family { ipv4 unicast | ipv4 multicast | ipv4 labeled-unicast | ipv4 tunnel | ipv4 mdt | ipv6 unicast | ipv6 multicast | ipv6 labeled-unicast | vpnv4 unicast | vpvn6 unicast }
5. route-policy policy-name { in | out }
6. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

**Command or Action**

**Step 2**

`router bgp as-number`

**Example:**

```
RP/0/RSP0/CPU0:router(config)# router bgp 125
```

**Purpose**

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/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.20
```

**Purpose**

Specifies a neighbor IP address.

**Step 4**

`address-family { ipv4 unicast | ipv4 multicast | ipv4 labeled-unicast | ipv4 tunnel | ipv4 mdt | ipv6 unicast | ipv6 multicast | ipv6 labeled-unicast | vpng unicast | vpng6 unicast }

**Example:**

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
```

**Purpose**

Specifies the address family.

**Step 5**

`route-policy policy-name { in | out }

**Example:**

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy example1 in
```

**Purpose**

Attaches the route-policy, which must be well formed and predefined.

**Step 6**

`commit`

---

### Modifying a Routing Policy Using a Text Editor

This task explains how to modify an existing routing policy using a text editor. See Editing Routing Policy Configuration Elements, on page 484 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Identifies the route policy, prefix set, AS path set, community set, or extended community set name to be modified.</td>
</tr>
<tr>
<td><strong>edit</strong></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>After editing with Nano, save the editor buffer and exit the editor by using the Ctrl-X keystroke.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

**Example:**
```
RP/0/RSP0/CPU0:router# edit route-policy sample1
```

| **Step 2** | (Optional) Displays the configuration of a specific named route policy. |
| **show rpl 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. |
| **name** | To display the contents of a named AS path set, community set, or extended community set, replace the **prefix-set** keyword with **as-path-set**, **community-set**, or **extcommunity-set**, respectively. |

**Example:**
```
RP/0/RSP0/CPU0:router# show rpl route-policy sample2
```

| **Step 3** | (Optional) Displays the contents of a named prefix set. |
| **show rpl prefix-set** | To display the contents of a named AS path set, community set, or extended community set, replace the **prefix-set** keyword with **as-path-set**, **community-set**, or **extcommunity-set**, respectively. |
| **name** | **Example:**
| **states** |```
RP/0/RSP0/CPU0:router# show rpl prefix-set prefixset1
```

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.

```con
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.

```con
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
end-policy
!
Using Wildcards For Routing Policy Sets: Example

This section describes examples of configuring routing policy sets with wildcards.

Using 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/0/0/CPU0:ios(config)# prefix-set pfx_set1
   RP/0/0/CPU0:ios(config-pfx)# 1.2.3.4/32
   RP/0/0/CPU0:ios(config-pfx)# end-set
   RP/0/0/CPU0:ios(config)# prefix-set pfx_set2
   RP/0/0/CPU0:ios(config-pfx)# 2.2.2.2/32
   RP/0/0/CPU0:ios(config-pfx)# end-set
   ```

2. Configure a route policy with wildcards to refer to the prefix sets.
   
   ```
   RP/0/0/CPU0:ios(config)# route-policy WILDCARD_PREFIX_SET
   RP/0/0/CPU0:ios(config-rpl)# if destination in prefix-set* then pass else drop endif
   RP/0/0/CPU0:ios(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/0/0/CPU0:ios(config)# commit
   ```

   This completes the configuration of routing policy with wildcards for prefix sets. For detailed information on prefix sets, see prefix-set, on page 430.

Using 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/0/0/CPU0:ios(config)# as-path-set AS_SET1
   RP/0/0/CPU0:ios(config-as)# ios-regex '_22$'
   RP/0/0/CPU0:ios(config-as)# ios-regex '_25$'
   RP/0/0/CPU0:ios(config-as)# end-set
   RP/0/0/CPU0:ios(config)# as-path-set AS_SET2
   RP/0/0/CPU0:ios(config-as)# ios-regex '_42$'
   RP/0/0/CPU0:ios(config-as)# ios-regex '_47$'
   RP/0/0/CPU0:ios(config-as)# end-set
   ```

2. Configure a route policy with wildcards to refer to the AS-path sets.
   
   ```
   RP/0/0/CPU0:ios(config)# route-policy WILDCARD_AS_SET
   RP/0/0/CPU0:ios(config-rpl)# if as-path in as-path-set* then pass else drop endif
   RP/0/0/CPU0:ios(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.
Using 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/0/0/CPU0:ios(config)# community-set CSET1
   RP/0/0/CPU0:ios(config-comm)# 12:24, 12:36, 12:72
   RP/0/0/CPU0:ios(config-comm)# end-set
   RP/0/0/CPU0:ios(config)# community-set CSET2
   RP/0/0/CPU0:ios(config-comm)# 24:12, 24:42, 24:64
   RP/0/0/CPU0:ios(config-comm)# end-set

2. Configure a route policy with wildcards to refer to the community sets.

   RP/0/0/CPU0:ios(config)# route-policy WILDCARD_COMMUNITY_SET
   RP/0/0/CPU0:ios(config-rpl)# if community matches-any community-set* then pass else drop endif
   RP/0/0/CPU0:ios(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/0/0/CPU0:ios(config)# commit

   This completes the configuration of routing policy with wildcards for community sets. For detailed information on community path sets, see community-set, on page 426.

Using Wildcards for Extended Community Sets

Use the following example to configure a routing policy with wildcards for extended community sets.

1. Configure the required extended community sets in the global configuration mode.

   RP/0/0/CPU0:ios(config)# extcommunity-set rt RT_SET1
   RP/0/0/CPU0:ios(config-ext)# 1.2.3.4:555, 1234:555
   RP/0/0/CPU0:ios(config-ext)# end-set
   RP/0/0/CPU0:ios(config)# extcommunity-set rt RT_SET2
   RP/0/0/CPU0:ios(config-ext)# 1.1.1.1:777, 1111:777
   RP/0/0/CPU0:ios(config-ext)# end-set

2. Configure a route policy with wildcards to refer to the extended community sets.

   RP/0/0/CPU0:ios(config)# route-policy WILDCARD_EXT_COMMUNITY_SET
   RP/0/0/CPU0:ios(config-rpl)# if extcommunity rt matches-any extcommunity-set* then pass else drop endif
   RP/0/0/CPU0:ios(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/0/0/CPU0:ios(config)# commit
This completes the configuration of routing policy with wildcards for extended community sets. For detailed information on extended community path sets, see `extcommunity-set`, on page 427.

### Additional References

The following sections provide references related to implementing RPL.

#### Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing policy language commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td>Routing Policy Language Commands on Cisco ASR 9000 Series Router module of the Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference</td>
</tr>
<tr>
<td>Regular expression syntax</td>
<td>Understanding Regular Expressions, Special Characters and Patterns appendix in the Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide</td>
</tr>
</tbody>
</table>

#### Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>

#### MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>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: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></td>
</tr>
</tbody>
</table>

#### RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 1771</td>
<td>A Border Gateway Protocol 4 (BGP-4)</td>
</tr>
<tr>
<td>RFC 4360</td>
<td>BGP Extended Communities Attribute</td>
</tr>
</tbody>
</table>
### Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a></td>
</tr>
</tbody>
</table>
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 514 section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the Cisco ASR 9000 Series Aggregation Services Router Commands Master List.

Feature History for Implementing Static Routes

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.7.2</td>
<td>This feature was introduced.</td>
</tr>
<tr>
<td>Release 4.0.1</td>
<td>The Dynamic ECMP Support for IGP Prefixes feature was added.</td>
</tr>
<tr>
<td>Release 4.2.1</td>
<td>The Enhanced Object Tracking for IP Static feature was added.</td>
</tr>
</tbody>
</table>

- Prerequisites for Implementing Static Routes, page 502
- Restrictions for Implementing Static Routes, page 502
- Information About Implementing Static Routes, page 502
- How to Implement Static Routes, page 505
- Configuration Examples, page 512
- Additional References, page 513
Prerequisites for Implementing Static Routes

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Restrictions for Implementing Static Routes

These restrictions apply while implementing Static Routes:

- Static routing to an indirect next hop, (any prefix learnt through the RIB and may be more specific over the AIB), that is part of a local subnet requires configuring static routes in the global table indicating the egress interfaces as next hop. To avoid forward drop, configure static routes in the global table indicating the next-hop IP address to be the next hop.

- Generally, a route is learnt from the AIB in the global table and is installed in the FIB. However, this behavior will not be replicated to leaked prefixes. Because the AIB from the global table is not present in the VRF, the leaked FIB entry takes reference from the RIB rather than the same view as the global table, which also relies on the AIB. This could lead to inconsistencies in forwarding behavior.

Information About Implementing Static Routes

To implement static routes you need to understand the following concepts:

Static Route Functional Overview

Networking devices forward packets using route information that is either manually configured or dynamically learned using a routing protocol. Static routes are manually configured and define an explicit path between two networking devices. Unlike a dynamic routing protocol, static routes are not automatically updated and must be manually reconfigured if the network topology changes. The benefits of using static routes include security and resource efficiency. Static routes use less bandwidth than dynamic routing protocols, and no CPU cycles are used to calculate and communicate routes. The main disadvantage to using static routes is the lack of automatic reconfiguration if the network topology changes.

Static routes can be redistributed into dynamic routing protocols, but routes generated by dynamic routing protocols cannot be redistributed into the static routing table. No algorithm exists to prevent the configuration of routing loops that use static routes.

Static routes are useful for smaller networks with only one path to an outside network and to provide security for a larger network for certain types of traffic or links to other networks that need more control. In general, most networks use dynamic routing protocols to communicate between networking devices but may have one or two static routes configured for special cases.

Default Administrative Distance

Static routes have a default administrative distance of 1. A low number indicates a preferred route. By default, static routes are preferred to routes learned by routing protocols. Therefore, you can configure an administrative
distance with a static route if you want the static route to be overridden by dynamic routes. For example, you
could have routes installed by the Open Shortest Path First (OSPF) protocol with an administrative distance
of 120. To have a static route that would be overridden by an OSPF dynamic route, specify an administrative
distance greater than 120.

**Directly Connected Routes**

The routing table considers the static routes that point to an interface as “directly connected.” Directly connected
networks are advertised by IGP routing protocols if a corresponding `interface` command is contained under
the router configuration stanza of that protocol.

In directly attached static routes, only the output interface is specified. The destination is assumed to be directly
attached to this interface, so the packet destination is used as the next hop address. The following example
shows how to specify that all destinations with address prefix 2001:0DB8::/32 are directly reachable through
interface GigabitEthernet 0/5/0/0:

```bash
RP/0/RSP0/CPU0:router(config)# router static
RP/0/RSP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RSP0/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:

```bash
RP/0/RSP0/CPU0:router(config)# router static
RP/0/RSP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RSP0/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:

```bash
RP/0/RSP0/CPU0:router(config)# router static
RP/0/RSP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RSP0/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/RSP0/CPU0:router(config)# router static
RP/0/RSP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RSP0/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/RSP0/CPU0:router(config)# router static
RP/0/RSP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RSP0/CPU0:router(config-static-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 210
```

Any of the three types of static routes can be used as a floating static route. A floating static route must be configured with an administrative distance that is greater than the administrative distance of the dynamic routing protocol because routes with smaller administrative distances are preferred.

---

**Note**

By default, static routes have smaller administrative distances than dynamic routes, so static routes are preferred to dynamic routes.

**Default VRF**

A static route is always associated with a VPN routing and forwarding (VRF) instance. The VRF can be the default VRF or a specified VRF. Specifying a VRF, using the `vrf vrf-name` command, allows you to enter VRF configuration mode for a specific VRF where you can configure a static route. If a VRF is not specified, a default VRF static route is configured.

**IPv4 and IPv6 Static VRF Routes**

An IPv4 or IPv6 static VRF route is the same as a static route configured for the default VRF. The IPv4 and IPv6 address families are supported in each VRF.
Dynamic ECMP

The dynamic ECMP (equal-cost multi-path) for IGP (Interior Gateway Protocol) prefixes feature supports dynamic selection of ECMP paths ranging from 1 to 32 IGP paths. ECMP for non-recursive prefixes is dynamic. This feature enables loadbalancing support in hardware among egress links.

Cisco ASR 9000 Series Router supports 32 IGP dynamic ECMP paths and 32 LDP (Label Distribution Protocol) dynamic ECMP paths.

This feature enables loadbalancing support in hardware among egress links.

Note

8-32 ECMP paths are available for BGP recursive prefixes. The ASR 9000 Enhanced Ethernet Line Card supports 32 ECMP paths for BGP prefixes and the ASR 9000 Ethernet Line Card supports 8 ECMP paths for BGP prefixes.

How to Implement Static Routes

This section contains the following procedures:

Configure Static Route

Static routes are entirely user configurable and can point to a next-hop interface, next-hop IP address, or both. In the software, if an interface was specified, then the static route is installed in the Routing Information Base (RIB) if the interface is reachable. If an interface was not specified, the route is installed if the next-hop address is reachable. The only exception to this configuration is when a static route is configured with the permanent attribute, in which case it is installed in RIB regardless of reachability.

Note

Currently, only default VRF is supported. VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families will be supported in a future release.

This task explains how to configure a static route.

SUMMARY STEPS

1. configure
2. router static
3. vrf vrf-name
4. address-family { ipv4 | ipv6 } { unicast | multicast }
5. prefix mask [vrf vrf-name] { ip-address | interface-type interface-instance } [ distance ] [ description text ] [ tag tag ] [ permanent ]
6. commit
**DETAILED STEPS**

**Step 1** configure

**Step 2** router static

**Example:**

```
RP/0/RSP0/CPU0:router(config)# router static
```

Enters static route configuration mode.

**Step 3** vrf vrf-name

**Example:**

```
RP/0/RSP0/CPU0:router(config-static)# vrf vrf_A
```

(Optional) Enters VRF configuration mode.
If a VRF is not specified, the static route is configured under the default VRF.

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

**Example:**

```
RP/0/RSP0/CPU0:router(config-static-vrf)# address family ipv4 unicast
```

Enters address family mode.

**Step 5** prefix mask [ vrf vrf-name ] { ip-address | interface-type interface-instance } [ distance ] [ description text ] [ tag tag ] [ permanent ]

**Example:**

```
RP/0/RSP0/CPU0:router(config-static-vrf-afi)# 10.0.0.0/8 172.20.16.6 110
```

Configures an administrative distance of 110.

• This example shows how to route packets for network 10.0.0.0 through to a next hop at 172.20.16.6 if dynamic information with administrative distance less than 110 is not available.

**Step 6** commit

A default static route is often used in simple router topologies. In the following example, a route is configured with an administrative distance of 110.

```
configure
router static
   address-family ipv4 unicast
      0.0.0.0/0 2.6.0.1 110
end
```

**Configure Floating Static Route**

This task explains how to configure a floating static route.
SUMMARY STEPS

1. configure
2. router static
3. vrf vrf-name
4. address-family { ipv4 | ipv6 } { unicast | multicast }
5. prefix mask [ vrf vrf-name ] { ip-address | interface-type interface-instance } [ distance ] [ description text ] [ tag tag ] [ permanent ]
6. commit

DETAILED STEPS

Step 1 configure
Step 2 router static
   
   Example:
   
   RP/0/RSP0/CPU0:router(config)# router static
   
   Enters static route configuration mode.

Step 3 vrf vrf-name
   
   Example:
   
   RP/0/RSP0/CPU0:router(config-static)# vrf vrf_A
   
   (Optional) Enters VRF configuration mode.
   
   If a VRF is not specified, the static route is configured under the default VRF.

Step 4 address-family { ipv4 | ipv6 } { unicast | multicast }
   
   Example:
   
   RP/0/RSP0/CPU0:router(config-static-vrf)# address family ipv6 unicast
   
   Enters address family mode.

Step 5 prefix mask [ vrf vrf-name ] { ip-address | interface-type interface-instance } [ distance ] [ description text ] [ tag tag ] [ permanent ]
   
   Example:
   
   RP/0/RSP0/CPU0:router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
   
   Configures an administrative distance of 201.

Step 6 commit
A floating static route is often used to provide a backup path if connectivity fails. In the following example, a route is configured with an administrative distance of 201.

```configure
router static
address-family ipv6 unicast
2001:0DB8::/32 2001:0DB8:3000::1 201
end
```

# Configure Static Routes Between PE-CE Routers

This task explains how to configure static routing between PE-CE routers.

---

**Note**
VRF fallback is not supported with IPv6 VPN Provider Edge (6VPE).

---

**SUMMARY STEPS**

1. `configure`
2. `router static`
3. `vrf vrf-name`
4. `address-family {ipv4 | ipv6} {unicast | multicast}`
5. `prefix mask [vrf vrf-name] {ip-address | interface-type interface-path-id} [distance] [description text] [tag tag] [permanent]`
6. `commit`

**DETAILED STEPS**

**Step 1**

`configure`

**Example:**

```
RP/0/RSP0/CPU0:router(config)# router static
```

Enters static route configuration mode.

**Step 2**

`router static`

**Example:**

```
RP/0/RSP0/CPU0:router(config)# router static
```

Enters static route configuration mode.

**Step 3**

`vrf vrf-name`

**Example:**

```
RP/0/RSP0/CPU0:router(config-static)# vrf vrf_A
```

(Optional) Enters VRF configuration mode.

If a VRF is not specified, the static route is configured under the default VRF.

**Step 4**

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

```
RP/0/RSP0/CPU0:router(config-static-vrf)# address family ipv6 unicast
Enters address family mode.
```

Step 5

```
prefix mask [vrf vrf-name ] [ ip-address | interface-type interface-path-id ] [ distance ][ description text ] [ tag tag ] [ permanent ]
```

Example:

```
RP/0/RSP0/CPU0:router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
Configures an administrative distance of 201.
```

Step 6

```
commit
```

In the following example, a static route between PE and CE routers is configured, and a VRF is associated with the static route:

```
configure
router static
vrf vrf_A
address-family ipv4 unicast
0.0.0.0/0 2.6.0.2 120
end
```

**Change Maximum Number of Allowable Static Routes**

This task explains how to change the maximum number of allowable static routes.

**Before You Begin**

![Note]

The number of static routes that can be configured on a router for a given address family is limited by default to 4000. The limit can be raised or lowered using the **maximum path** command. Note that if you use the **maximum path** command to reduce the configured maximum allowed number of static routes for a given address family below the number of static routes currently configured, the change is rejected. In addition, understand the following behavior: If you commit a batch of routes that would, when grouped, push the number of static routes configured above the maximum allowed, the first \( n \) routes in the batch are accepted. The number previously configured is accepted, and the remainder are rejected. The \( n \) argument is the difference between the maximum number allowed and number previously configured.

**SUMMARY STEPS**

1. **configure**
2. **router static**
3. **maximum path** { ipv4 | ipv6 } value
4. **commit**
DETAILED STEPS

Step 1 configure
Step 2 router static

Example:
RP/0/RSP0/CPU0:router(config)# router static
Enters static route configuration mode.

Step 3 maximum path {ipv4 | ipv6} value

Example:
RP/0/RSP0/CPU0:router(config-static)# maximum path ipv4 10000
Changes the maximum number of allowable static routes.
- Specify IPv4 or IPv6 address prefixes.
- Specify the maximum number of static routes for the given address family. The range is from 1 to 140000.
- This example sets the maximum number of static IPv4 routes to 10000.

Step 4 commit

Configuring a static route to point at interface null 0 may be used for discarding traffic to a particular prefix. For example, if it is required to discard all traffic to prefix 2001:0DB8:42:1/64, the following static route would be defined:

```plaintext
configure
router static
address-family ipv6 unicast
2001:0DB8:42:1::/64 null 0
end
```

Associate VRF with a Static Route

This task explains how to associate a VRF with a static route.

SUMMARY STEPS

1. configure
2. router static
3. vrf vrf-name
4. address-family {ipv4 | ipv6} {unicast | multicast}
5. prefix mask [vrf vrf-name] {next-hop ip-address | interface-name} {path-id} [distance] [description text] [tag tag] [permanent]
6. commit
DETAILED STEPS

Step 1  configure
Step 2  router static

Example:
RP/0/RSP0:/CPU0:router(config)# router static
Enters static route configuration mode.

Step 3  vrf  vrf-name

Example:
RP/0/RSP0:/CPU0:router(config-static)# vrf vrf_A
Enters VRF configuration mode.

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

Example:
RP/0/RSP0:/CPU0:router(config-static-vrf)# address family ipv6 unicast
Enters address family mode.

Step 5  prefix mask  [ vrf  vrf-name ]  { next-hop  ip-address | interface-name }  { path-id }  { distance }  { description  text }  [ tag  tag ]  [ permanent ]

Example:
RP/0/RSP0:/CPU0:router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
Configures an administrative distance of 201.

Step 6  commit

Enabling Object Tracking for Static Routes

Perform this task to enable object tracking for static routes.
SUMMARY STEPS

1. configure
2. router static
3. Use one of these commands:
   • address-family {ipv4} unicast
   • vrf vrf-name address-family {ipv4} unicast

4. ip-address / length [vrf vrf-name] {next-hop | type interface-path-id} track object-name
5. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters static route configuration mode.</td>
</tr>
<tr>
<td>Step 2 router static</td>
<td>Enters the address family or VRF address family configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)#router static</td>
<td></td>
</tr>
<tr>
<td>Step 3 Use one of these commands:</td>
<td></td>
</tr>
<tr>
<td>• address-family {ipv4} unicast</td>
<td></td>
</tr>
<tr>
<td>• vrf vrf-name address-family {ipv4} unicast</td>
<td></td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-static)#address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td>Or</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-static)#vrf vrf_1 address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td>Step 4 ip-address / length [vrf vrf-name] {next-hop</td>
<td>type interface-path-id} track object-name</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-static-afi)#100.0.24.0/24 204.0.23.2 track object1</td>
<td></td>
</tr>
<tr>
<td>Step 5 commit</td>
<td></td>
</tr>
</tbody>
</table>

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
```

Configuring a Static Route Between PE-CE Routers: Example

In the following example, a static route between PE and CE routers is configured, and a VRF is associated with the static route:

```
configure
router static
vrf vrf_A
address-family ipv4 unicast
0.0.0.0/0 2.6.0.2 120
end
```

Additional References

The following sections provide references related to implementing Static Routes.
### Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static routes commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td>Static Routing Commands in Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference</td>
</tr>
<tr>
<td>MPLS Layer 3 VPN configuration: configuration concepts, task, and examples</td>
<td>Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide</td>
</tr>
</tbody>
</table>

### Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>

### MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>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: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></td>
</tr>
</tbody>
</table>

### RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>

### Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a></td>
</tr>
</tbody>
</table>
Implementing RCMD

This module describes how to implement RCMD.

Feature History for Implementing RCMD

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 4.2.0</td>
<td>This feature was introduced.</td>
</tr>
</tbody>
</table>

- Route Convergence Monitoring and Diagnostics, page 517
- Configuring Route Convergence Monitoring and Diagnostics, page 518

Route Convergence Monitoring and Diagnostics

Route Convergence Monitoring and Diagnostics (RCMD) is a mechanism to monitor OSPF and ISIS convergence events, gather details about the SPF runs and time taken to provision routes and LDP labels across all LCs on the router.

RCMD is a tool that collects and reports data related to routing convergence. Highlights of the RCMD mechanism are:

- Lightweight and always-on using route flow markers across routing components (all nodes & MC).
- Tracks most convergence events and all routes affected by them.
- Provides within-router view with statistics and time-lines on per convergence event basis.
- Measurements against time-line/SLA and triggers specified EEM actions on excess.
- 'On the router' reports via CLI/XML interface.
- Each RCMD enabled router provides a digest of convergence data.

The events that are monitored and reported by RCMD are:

- OSPF and IS-IS SPF events (default VRF only).
- Add/delete of specific external or inter-area/level prefixes.
• IGP flooding propagation delays for LSA/LSP changes.

RCMD runs in two modes:
• Monitoring—detecting events and measuring convergence.
• Diagnostics—additional (debug) information collection for 'abnormal' events.

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router-convergence</td>
<td>Enters configure Router Convergence Monitoring and Diagnostics (rcmd) configuration mode.</td>
</tr>
</tbody>
</table>

Example: RP/0/RSP0/CPUD0:router(config)#router-convergence
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>collect-diagnostics location</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-rcmd)#collect-diagnostics 0/3/CPU0</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>event-buffer-size number</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-rcmd)#event-buffer-size 100</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>max-events-stored number</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-rcmd)#max-events-stored 10</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>monitoring-interval minutes</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-rcmd)#monitoring-interval 120</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>node node-name</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-rcmd)#node</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>protocol</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-rcmd)#protocol ISIS</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-rcmd proto)#</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td><strong>priority</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-rcmd proto)#priority critical</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-rcmd proto-prio)#</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td><strong>disable</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-rcmd proto-prio)#disable</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>Enables leaf network monitoring. Specify a maximum number of leaf networks to be monitored. Range for maximum number is 10-100.</td>
</tr>
<tr>
<td>leaf-network <em>number</em></td>
<td>Example: RP/0/RSP0/CPU0:router(config-rcmd-proto-prio)#leaf-network 100</td>
</tr>
<tr>
<td><strong>Step 12</strong></td>
<td>Specifies threshold value for convergence in milliseconds. Select a threshold value from the range. Range is 0-4294967295 milliseconds</td>
</tr>
<tr>
<td>threshold <em>value</em></td>
<td>Example: RP/0/RSP0/CPU0:router(config-rcmd-proto-prio)#threshold 1000</td>
</tr>
<tr>
<td><strong>Step 13</strong></td>
<td>Sets the absolute directory path for storing diagnostic reports.</td>
</tr>
<tr>
<td>storage-location</td>
<td>Example: RP/0/RSP0/CPU0:router(config-rcmd)#storage-location RP/0/RSP0/CPU0:router(config-rcmd-store)#</td>
</tr>
<tr>
<td><strong>Step 14</strong></td>
<td>Specifies the absolute directory path for storing diagnostic reports. Set a directory-path-name. Example: /disk0:/rcmd or &lt;tftp-location&gt;/rcmd/ diagnostics</td>
</tr>
<tr>
<td>diagnostics <em>directory-path-name</em></td>
<td>Example: RP/0/RSP0/CPU0:router(config-rcmd-store)#diagnostics /disk0:/rcmd</td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td>Specify a maximum size for the diagnostics directory. Set the size in %. Range is 5%-80%.</td>
</tr>
<tr>
<td>diagnostics-size</td>
<td>Example: RP/0/RSP0/CPU0:router(config-rcmd-store)# diagnostics-size 8</td>
</tr>
<tr>
<td><strong>Step 16</strong></td>
<td>Specifies the absolute directory path for storing reports. Set a directory-path-name. Example: /disk0:/rcmd or &lt;tftp-location&gt;/rcmd/</td>
</tr>
<tr>
<td>reports <em>directory-path-name</em></td>
<td>Example: RP/0/RSP0/CPU0:router(config-rcmd-store)#reports /disk0:/rcmd</td>
</tr>
<tr>
<td><strong>Step 17</strong></td>
<td>Specify a maximum size for the reports directory. Set the size in %. Range is 5%-80%.</td>
</tr>
<tr>
<td>reports-size</td>
<td>Example: RP/0/RSP0/CPU0:router(config-rcmd-store)#reports-size 8</td>
</tr>
</tbody>
</table>
INDEX

A
ABRs (area border routers) 295
accumulated interior gateway protocol (aigp) 52
action 443
additional-path attach point 446
address family command 13, 233
address-family ipv4 command 160, 161, 172, 173
adjacencies, tuning 264
adjacency 308
adjacency (OSPFv2) 298
administrative distance 383, 402, 502
forwarding, static routes 502
aggregation 447
as-path-set, inline set form 425
as-path-set, named set form 425
ASBRs (autonomous system boundary routers) 296
attached 484
attached bit on an instance 239
attached bit on an IS-IS instance 239
attaching to BGP neighbor 491
attributes 435, 437
authentication 297, 298, 322
configuring (OSPFv2) 322
MD5 (OSPFv2) 297
route, key rollover (OSPFv2) 298
strategies 297
authentication keychain configuration 224
authentication using keychain 210
authentication using keychain in RIP 403
authentication, configuring 260, 322
autonomous system number format 12
autonomous systems 294

B
backbone area 295
bandwidth 409
benefits 400
bestpath algorithm 36
BFD 151, 152, 154, 160, 161, 162, 167, 168, 169, 170, 171, 172, 173, 179, 180, 181, 183, 184, 185, 186, 187, 188
BGP configuration mode 167
echo mode, disabling 183, 184
fast detection, configuring 171, 172
IPv6 162
IPv6 checksum, enabling and disabling 186, 187
OSPF 169
configuration mode 169
OSPFv3 configuration mode 171
setting 167, 168, 169, 171
minimum interval 167, 168, 171
VLAN bundles 162
BFD configuration mode 179, 180, 181, 184, 185, 186, 187
counters 188
clearing 188
displaying 188
echo mode, disabling 183, 184
enabling 160, 167, 168, 170, 172
interface 168, 170
neighbor 167
static route 160, 172
IPv6 checksum, enabling and disabling 186, 187
overview 152
prerequisites 151
setting 167, 168, 169, 171
minimum interval 167, 168, 171
multiplier 169, 171
source and destination ports 154
static routes, configuring 160, 161, 172, 173
bfd command 179, 180, 181, 184, 185, 186, 187
bfd fast-detect command 167, 168, 169, 170, 171, 172
bfd minimum-interval command 167, 168, 169, 171
BFD multipath support for BGP 58
bfd multiplier command 167, 168, 169, 171
Index

label-allocation-mode per-ce command 112, 113
redistribute connected command 116, 117
export route-policy command 104, 105
bgp vrf address family submode 14
import route-policy command 104, 105
maximum prefix command 104, 105
import route-target command 104, 105
ebgp-multihop command 112, 114
as-override command 112, 115
site-of-origin command 112, 114
allowas-in command 112, 115
bgp VRF neighbor address family submode 17
bgp VRF neighbor submode 17
bgp vrf submode 13
bidirectional forwarding detection 7
Boolean operator precedence 437
Boolean operators, types 444

c
CIDR 400
Cisco IOS and Cisco IOS XR software differences, configuration 233
grouped 233
Cisco IOS XR OSPFv3 and OSPFv2 differences 292
clear policy 456
CLI (command-line interface) inheritance 293
CLI inheritance 293
community-set, inline set form 426
community-set, named set form 426
components 431
configuration 18, 20, 231, 233, 239, 243, 245, 254, 318, 335, 339, 345
grouped configuration 233
grouping 18
inheriting 20
inheriting templates 20
Level 1 or Level 2 routing 243
MPLS TE 339
multitopology 254
neighbors, nonbroadcast networks 318
restrictions 231
sham-links 345
single topology 245
SPF throttling 335
configuration and operation, verifying 364
configuration basics 434
configuration elements, editing 484
configuration, grouped 233
configurations (BGP) 20
configuring 127, 239, 254, 258, 333, 343, 356
graceful restart 343
configuring (OSPFv2) 322
configuring authentication 261
configuring bgp additional paths 130
connected 503
Constrained route distribution 54
control or prevent routing updates 409
controlling 254
controlling frequency 325
controlling the frequency 325
cost community, BGP 31
creating 327
customizing (IS-IS) 269
customizing routes 269
d
Dampening 448
dampening (BFD) command 185
dampening, route 43
data structures in BGP and other protocols 382
debug 456
default 239, 298
IS-IS 239
OSPFv2 298
default address family 28, 48
default drop disposition 438
default originate 448, 470, 474, 476
default route 298
default routes 239
default-accept-in 477
default-accept-out 478
default-information originate 481
defining 490
definitions 434
deploying 386
description 5, 30, 289, 305, 306, 381
designate router (DR) 298
Designate Router (DR) 298
designated router (DR) 298
Designated Route (DR) 298
displaying information 344
disposition 441
dynamic ECMP for IGP prefixes 505
e
Echo disable command 184, 185
echo ipv4 source command 179, 180, 181, 184
echo latency detect command 181
EIGRP 210, 224
authentication keychain configuration 224
authentication using keychain 210
EIGRP (Enhanced Interior Gateway Routing Protocol) 198, 199, 205, 206, 207, 217, 477, 478, 479
features 199
hello interval and hold time 206
overview 198
policy attach points 478, 479
default-accept-out 478
if-policy-in 479
if-policy-out 479
policy-in 478
policy-out 478
redistribute 479
policy attach points, default-accept-in 477
restrictions 217
routing policy options 207
split horizon 205
stub routing 206
elseif 443
enabling 243, 314
enabling multicast-intact 272, 351
enforcing, BGP 28
exabgp multipath loadsharing configuration 142
show eigrp traffic command 222, 224
clear eigrp neighbors command 222, 223
show isis spf-log command 268, 269
show eigrp interfaces command 222, 223
EXEC mode 23, 24, 26
show bgp af-group command 24
show bgp inheritance command 23
show bgp neighbor command 23
show bgp neighbor-group command 26
edit command 492, 493
show eigrp topology command 222, 224
clear eigrp topology command 222, 223
show rpl route-policy command 492, 493
show isis mpls command 263, 264
clear ospf command 365, 366
show isis command 244, 245
export 453
extended community set, inline form 427
extended community set, named form 427

F
features 199
filter network updates 409
floating 504
functional overview 7, 291, 382

G
Generalized TTL Security Mechanism (GTSM), TTL value 312
Generalized TTL Security Mechanism (GTSM), configuring virtual links 312
global parameters 436
global-inbound 482
global-inbound, policy attach points 482
graceful restart 306, 343
graceful-restart helper command 307
graceful-restart interval command 307
graceful-restart lifetime command 307
grouped 233
grouped configuration 233
grouping 18

H
hello interval and hold time 206
hop count 400

I
ibgp multipath load sharing 131
iBGP Multipath Load Sharing 52
if 443
if-policy-in 479
if-policy-out 479
implementing 422
prerequisites 422
implementing in 422
prerequisites 422
import 452
inheritance 20, 23
configurations (BGP) 20
monitoring 23
inheritance, monitoring 23
inheriting 20
inheriting templates 20
instance and router ID 296
inter-area-propagate 476
interface attributes and limitations 310
interface command 211, 212, 314, 315, 406
interface-inbound 482
interface-outbound 482
interior routers 296
IP fast reroute 242, 276
IPv4 and IPv6 support 383
ipv4 bgp-policy accounting 53
IPv6 236, 383
  IS-IS support 236
    single-topology 236
  RIB support 383
IPv6 and IPv6 VPN provider edge transport 384
IPv6 and IPv6 VPN provider edge transport over MPLS 384
ipv6 checksum command 186, 187, 188
IPv6 support 236
ipv6 unicast routing 53
ipv6 uRPF 54
IS-IS 239
  configuration, grouped 233
  adjacencies, tuning 264
  attached bit on an instance 239
  authentication, configuring 260
Cisco IOS and Cisco IOS XR software differences, configuration 233
  grouped 233
  configuration 231, 233, 243, 245, 254
    grouped configuration 233
    Level 1 or Level 2 routing 243
    multitopology 254
    restrictions 231
    single topology 245
    configuring authentication 261
    customizing routes 269
    default routes 239
    enabling 243
    enabling multicast-intact 272
    grouped configuration 233
    IP fast reroute 276
    Level 1 or Level 2 routing, configuration 243
    LSP flooding 254
      controlling 254
    MPLS LDP IS-IS synchronization 271
    multi-instance IS-IS 238
    multitopology, configuring 254
    nonstop forwarding 237, 258
      configuring 258
    overload bit 238, 239
      configuring 239
      on router 238
    policy attach points 475, 476
      default originate 476
      inter-area-propagate 476
      redistribute 475
    priority for prefixes added to RIB 275
    restrictions, configuring 231
    set SPF interval 267
    single topology, configuring 245
    single topology, IPv6 support 236
  IS-IS (Intermediate System-to-Intermediate System) (continued)
    tagging IS-IS interface routes 273
    IS-IS address family submode 233
    IS-IS Overload Bit Avoidance 239
    IS-IS support 236
      single-topology 236
  K
    key rollover 298
    keychains 49
  L
    label consistency checker 385
    LDP IS-IS synchronization 271
    Level 1 or Level 2 routing 243
    Level 1 or Level 2 routing, configuration 243
    line card roles and filters in selective vrf download 56
    link-state advertisement (LSA) 299, 308
      OSPFv2 299
      OSPFv3 299, 308
    load balancing 310
    loop-free alternate 242
    LSA 299, 325, 331
      controlling frequency 325
      controlling the frequency 325
      on an OSPF ABR 331
      types 299
    LSP flooding 235, 254
      controlling 254
      on specific interfaces 235
    lsp-check-interval command 255, 256
  M
    maximum limits (BGP) 9
    maximum path command 509, 510
    MD5 297
    MD5 (OSPFv2) 297
    MD5 authentication 297
    Message Digest 5 (MD5) authentication 311
    message statistics 384
    metrics 400
    modification 437
    modifying 492
    monitor 126
    monitoring 23, 386
    MPLS 384
      IPv6 and IPv6 VPN provider edge transport 384
MPLS LDP IS-IS synchronization 271
MPLS TE 339
MPLS TE (Multiprotocol Label Switching traffic engineering) configuring 339
   OSPFv2 339
MPLS TE (Multiprotocol Label Switching traffic engineering), configuring for OSPFv2 339
MPLS TE, configuring 339
MPLS VPN carrier supporting carrier 48
multi-area adjacency 310, 356
   configuring 356
   interface attributes and limitations 310
   overview 310
multi-instance IS-IS 238
multicast topology 240
multicast-intact 240, 309
   OSPFv2 309
multiprotocol 41
multiprotocol BGP 41
Multiprotocol Label Switching (MPLS) 271
   LDP IS-IS synchronization 271
Multiprotocol Label Switching traffic engineering (MPLS TE), configuring for IS-IS 262
multitopology 254, 284
   (example) 284
   configuring 254
multitopology routing 249
multitopology, configuring 254

N
names 424
NBMA networks 297
neighbor address family command 13, 234
neighbor command 13, 233, 406
neighbor command (BFD) 167, 168
neighbor command (OSPFv2) 319, 321
   OSPFv3) 319, 321
neighbor export 449
neighbor import 449
neighbor-group command 18
neighbor-orf 455
neighbors 9, 298
   adjacency (OSPFv2) 298
   maximum limits (BGP) 9
neighbors, adjacency 298
neighbors, configuring nonbroadcast networks 318
neighbors, maximum limits on 9
neighbors, nonbroadcast networks 318
neighbors, nonbroadcast networks, configuring 318
   network 450
   next-hop 455
nonattached 484
nonstop forwarding 237, 258, 306, 337
   configuring 258
   description 306
not-so-stubby area 295

O
object tracking for static routes 511
on an OSPF ABR 331
on router 238
on specific interfaces 235
originating prefixes with AiGP 142
   example 142
   nssa command 316, 317
   network command 319, 320
   authentication message-digest command 328, 329
   virtual-link command 328, 329
   hello-interface (OSPF) command 319, 320
   default-cost command 316, 318
   dead interval command command 319, 320
OSPFv2 240, 298, 299, 309, 339
   multicast-intact 240
   authentication, configuring 322
Cisco IOS XR OSPFv3 and OSPFv2 differences 292
   CLI (command-line interface) inheritance 293
   configuration 318, 339, 345
   MPLS TE 339
   neighbors, nonbroadcast networks 318
   sham-links 345
   configuration and operation, verifying 364
   default route 298
   description 289
   designate router (DR) 298
   Designate Router (DR) 298
   enabling 314
   enabling multicast-intact 351
   functional overview 291
   instance and router ID 296
   LSA 298, 325, 331
      controlling the frequency 325
      on an OSPF ABR 331
      types 299
   MD5 authentication 297
   MPLS TE, configuring 339
   neighbors, adjacency 298
   neighbors, nonbroadcast networks, configuring 318
   nonstop forwarding 306, 337
      description 306
OSPFv2 (Open Shortest Path First Version 2) (continued)
policy attach points 470, 471, 474
  area-in 471
  area-out 471
default originate 470
redistribute 470, 474
route authentication methods 297, 298
  key rollover 298
MD5 297
plaintext 297
strategies 297
route redistribution 305, 333
  configuring 333
description 305
sham-link 345
Shortest Path First (SPF) throttling 305
description 305
Shortest Path First (SPF) throttling, configuring 335
SPF throttling, configuring 335
stub and not-so-stubby area types, configuring 316
supported OSPF network types 297
  NBMA networks 297
  point to point networks 297
  virtual link 301, 302
  creating 327
  transit area 301
OSPFv3 299, 306, 308
graceful restart 306
OSPFv3 (Open Shortest Path First Version 3) 289, 291, 292, 293,
  296, 298, 299, 301, 308, 310, 314, 316, 318, 325, 331, 333, 335, 343,
  364, 470, 474
Cisco IOS XR OSPFv3 and OSPFv2 differences 292
CLI inheritance 293
collection 335
  SPF throttling 335
collection and operation, verifying 364
configuring 343
graceful restart 343
default route 298
description 299
enabling 314
functional overview 291
instance and router ID 296
link-state advertisement (LSA) 308
load balancing 310
LSA 299, 325, 331
  controlling frequency 325
  on an OSPF ABR 331
types 299
neighbors, configuring nonbroadcast networks 318
policy attach points 470, 474
default originate 474
redistribute 470, 474
routes, redistribute 333

OSPFv3 (Open Shortest Path First Version 3) (continued)
  SPF (Shortest Path First) throttling configuring 335
  stub and not-so-stubby area types, configuring 316
  virtual link, description 301
OSPFv3 Graceful Restart feature 306, 308, 344, 345
  adjacency 308
displaying information 344, 345
OSPFv3 SPF 303
nsf command 258, 259, 407, 408
default-information originate command 269, 270
mpls traffic-eng router-id command 340
timers throttle SPF command 335, 336, 362, 363
mpls traffic-eng area command 340, 341
nsf interval command 338, 339
overload bit 238, 239
  configuration 239
  configuring 239
  on router 238
overview 198, 310, 423

P
parameterization 435
PCE extensions to OSPFv2 312
per ce label 53
per vrf label 53
PIM 240
plain text 297
point to point networks 297
point-to-point networks 297
policy 434, 435, 437, 438, 439, 441
  attributes 435, 437
  Boolean operator precedence 437
  configuration basics 434
  default drop disposition 438
  definitions 434
  statement processing 439
  statements, types 441
  verification 439
policy attach points 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 470,
  471, 474, 475, 476, 477, 478, 479, 481, 482
  aggregation 447
  allocate label 454
  area-in 471
  area-out 471
clear policy 456
dampening 448
debuge 456
default originate 448, 470, 474, 476
default-accept-in 477
default-accept-out 478
default-information originate 481
policy attach points (continued)
export 453
global-inbound 482
if-policy-in 479
if-policy-out 479
import 452
inter-area-propagate 476
interface-inbound 482
interface-outbound 482
neighbor export 449
neighbor import 449
neighbor-orf 455
network 450
next-hop 455
policy-in 478
policy-out 478
redistribute 450, 470, 474, 475, 479, 481
retain route target 454
show bgp 451
table policy 452

policy attach points, aggregation 447
policy attach points, default-accept-in 477

policy attributes 435, 437
parameterization 435
modification 437
policy-in 478
policy-out 478
policy, modifying 484
attached 484
nonattached 484
prefix prioritization 303
OSPFv2 SPF 303
prefix-set 430
prerequisites 382, 422
priority for prefixes added to RIB 275

Q
quarantining 384

R
redistribute 450, 470, 474, 475, 479, 481
redistribute (OSPFv2, OSPFv3) 333
redistributing (OSPFv2, OSPFv3) 333
redistribution 402
remark 441
remote-as command (BFD) 167, 168
restrictions 217, 231
restrictions, configuring 231
retain route target 454

RFC 2328, OSPF Version 2 292
RFC 2453, RIP Version 2 400
RFC 2740 OSPFv3 292
RFC 3682 312
Generalized TTL Security Mechanism (GTSM), TTL value 312
RIB 384
statistics 384
quarantining 384
RIB (Routing Information Base) 381, 382, 383, 386, 392
administrative distance 383
data structures in BGP and other protocols 382
deploying 386
description 381
examples 392
functional overview 382
IPv4 and IPv6 support 383
monitoring 386
prerequisites 382
RIB quarantining 384
RIB statistics 384
RIB support 383
RIP (Routing Information Protocol) 482
global-inbound, policy attach points 482
global-inbound 482
RIP (Routing Information Protocol) 400, 401, 402, 403, 409, 411, 481, 482
administrative distance 402
benefits 400
CIDR 400
control or prevent routing updates 409
metrics 400
redistribution 402
route policy creation 411
route timers 401
routing policy options 403
bandwidth 409
filter network updates 409
hop count 400
policy attach points 481, 482
default-information originate 481
global-inbound 482
interface-inbound 482
interface-outbound 482
redistribute 481
routing loops 409
split horizon, enabling IP 401
VLSMs (variable-length subnet masks) 400
WAN link 409
RIP v2 supported features 400
route authentication methods 297, 298
key rollover 298
MD5 297
route authentication methods (continued)

Plain text 297
strategies 297
route consistency checker 385
route dampening 43
end-policy command 68
route policy creation 411
route redistribution 305, 333
configuring 333
description 305
route redistribution (OSPFv2, OSPFv3) 305
route reflectors 44
route timers 401
set eigrp-metric command 216
route-policy pass-all command 28
route, key rollover (OSPFv2) 298
range command 331, 332
stub command 217, 218, 316, 317
distance command 211, 212
is-type command 244
net command 244
area command 314, 315
router bgp command 13, 167
router bgp neighbor group address family configuration mode, address family command 18
maximum-prefix command 214, 215
log-neighbor-warnings command 222, 223
default-metric command 211, 212
redistribute command 214, 215, 219, 220, 221
redistribute maximum-prefix 214, 215
log-neighbor-changes command 222, 223
maximum paths command 214, 215
timers nsf route-hold command 214, 215
router eigrp command 211
summary-address command 213, 214
holdtime command 211, 212
bandwidth-percent command 211, 212
router identifier 8
mpls traffic-engineering multicast-intact command 272, 273
set-attached-bit command 269, 271
single-topology command 246, 248
redistribute isis command 269, 270
metric-style wide command 273, 274, 275
spf-interval command 268
mpls traffic-eng command 263
ispf command 268, 269
spf prefix-priority command 275, 276
address-family (IS-IS) command 246, 248, 282, 283
sp-refresh-interval command 255
max-lsp-lifetime command 255, 256
ignore-lsp-errors command 255, 256
lsp-password command 260
lsp-mtu command 255, 256
lsp-gen-interval command 255, 256
nsf interface-expires command 258, 259
nsf interface-timer command 258, 259
retransmit-interval command 255, 257
hello-packet command 265, 266
mesh-group command 255, 257
hello-password command 260, 261, 265, 266
hello-interval (IS-IS) command 265, 266
hello-multiplier command 265, 266
csnp-interval command 255, 257
retransmit-throttle-interval command 255, 257
tag command 273, 274
router isis interface configuration submode 246
router ospf command 314, 315
router ospf command (BFD) 169
summary-prefix command 333, 334
See also router ospfv3 configuration submode
message-digest-key command 323, 324
authentication command (OSPFv2) 323, 324
router-id command 314, 315
timers ls refresh command 326
log adjacency changes command 314, 315, 362, 363
timers ls a gen-interval command 326
timers ls a group-pacing command 326, 327
timers ls a min-interval command 326, 327
router ospfv3 command 314, 315
router ospfv3 command (BFD) 171
router rib command 389
address-family command 389, 390
router rip command 406
output-delay command 407, 408
split-horizon disable command 407, 408
metric-zero-accept command 407, 408
broadcast-for-v2 command 406
poison-reverse command 407, 409
auto-summary command 407
receive version command 406
send version command 406
timers basic command 407, 408
passive-interface command 409, 410
router static command 509, 510
router static command (BFD) 160, 161, 172, 173
routes 239, 269, 284, 298, 333
customizing (IS-IS) 269
default 239, 298
IS-IS 239
OSPFv2 298
redistribute (OSPFv2, OSPFv3) 333
redistribute IS-IS routes (example) 284
redistribute IS-IS routes example 284
redistributing (OSPFv2, OSPFv3) 333
routes, redistribute 333
Index

routing components 294, 295, 296, 298
  ABRs (area border routers ) 295
  Area Border Routers (ABRs) 295
  ASBRs (autonomous system boundary routers) 296
  autonomous systems 294
  backbone area 295
  designated router (DR) 298
  Designated Router (DR) 298
  interior routers 296
  not-so-stubby area 295
  stub area 295
routing domain confederation 44
routing loops 409
routing policy 28, 422, 441, 443, 484, 490, 491, 492, 493, 494, 495
  attaching to BGP neighbor 491
  configuration elements, editing 484
  defining 490
  defining (example) 493
  enforcing, BGP 28
  implementing 422
    prerequisites 422
  implementing in 422
    prerequisites 422
  inbound (example) 494
  modifying 492
  modular inbound (example) 495
  statements 441, 443
    action 443
    disposition 441
    elseif 443
    if 443
    remark 441
Routing Policy 422
  restrictions, configuring routing policy 422
routing policy options 207, 403
routing policy, enforcing 28
RPL (routing policy language) 423, 424, 425, 426, 427, 430, 431, 434, 435, 437, 438, 439, 441, 444
  Boolean operators, types 444
  components 431
  overview 423
  policy 434, 435, 437, 438, 439, 441
    attributes 435, 437
    Boolean operator precedence 437
    configuration basics 434
    default drop disposition 438
    definitions 434
    statement processing 439
    statements, types 441
    verification 439
  policy attributes 435, 437
    parameterization 435
    modification 437
RPL (routing policy language) (continued)
  structure 424, 425, 426, 427, 430
    as-path-set, inline set form 425
    as-path-set, named set form 425
    community-set, inline set form 426
    community-set, named set form 426
    extended community set, inline form 427
    extended community set, named form 427
    names 424
    prefix-set 430
    sets 424
  RT constrained route distribution 54
    benefits 54
S
  security ttl command 362, 364
  selective vrf download 55
  set SPF interval 267
  set-overload-bit command 269, 270
  sets 424
  sham-link 345
  sham-links 345
  Shortest Path First (SPF) throttling 305
    description 305
  Shortest Path First (SPF) throttling, configuring 335
    show bgp 451
    show bgp af-group command 24
    show bgp cidr-only command 123
    show bgp command 123
    show bgp community command 123
    show bgp inheritance command 23
    show bgp ipv4 unicast summary command 125
    show bgp neighbor command 23
    show bgp neighbor-group command 26, 123, 124
    show bgp neighbors command 123, 124
    show bgp paths command 123, 124
    show bgp process command 125
    show bgp process detail command 125
    show bgp session-group command 26
    show bgp summary command 123, 124, 125
    show bgp vpnv4 unicast summary command 125
    show bgp vrf command 125
    show eigrp accounting command 222, 223
    show eigrp neighbors command 217, 218, 222, 224
    show isis database command 255, 257
    show isis database-log command 255, 258
    show isis interface command 265, 267
    show isis lsp-log command 255, 257
    show isis mpls traffic-eng adjacency-log command 263, 264
    show isis mpls traffic-eng advertisements command 263, 264
    show isis neighbors command 265, 267
show isis topology command 246, 249
show ospf command 328
show ospfv3 command 328
show placement program bgp command 125, 126
show placement program brb command 125, 126
show protocols eigrp command 222, 224
show running-config command 258, 259
single topology 245
single topology, configuring 245
single topology, IPv6 support 236
single-topology 236, 267, 283
configuring (example) 283
IPv6 support 236
set SPF interval 267
specified 504
SPF (Shortest Path First) throttling configuring 335
SPF throttling 335
SPF throttling, configuring 335
OSPFv2 (Open Shortest Path First Version 2) 335
split horizon 205
split horizon, enabling IP 401
statement processing 439
statements 441, 443
action 443
disposition 441
elseif 443
if 443
remark 441
statements, types 441
static routes 502, 503, 504
administrative distance 502
connected 503
floating 504
specified 504
statistics 384
strategies 297
structure 424, 425, 426, 427, 430
as-path-set, inline set form 425
as-path-set, named set form 425
community-set, inline set form 426
community-set, named set form 426
extended community set, inline form 427
extended community set, named form 427
names 424
prefix-set 430
sets 424
stub and not-so-stubby area types, configuring 316
stub area 295
stub area types, configuring (OSPFv3) 316
stub routing 206
router isis address family submode 269, 270
supported OSPF network types 297
NBMA networks 297
point to point networks 297

T
table policy 452
tagging IS-IS interface routes 273
transit area 301
transit area (OSPFv2) 301
types 299

U
unequal cost recursive load balancing 58
update groups 30, 31, 126
  BGP configuration 31
  BGP update generation 31
description 30
monitor 126

V
verification 439
virtual link 301, 327
  creating 327
  transit area 301
  transit area (OSPFv2) 301
virtual link, description 301
VLSMs (variable-length subnet masks) 400
VPNv4 address family command 17
VRF address family command 14
export route-target command 104, 105
VRF command 13
vrf command (BFD) 160, 161, 172, 173
VRF neighbor address family command 17
VRF neighbor command 17

W
WAN link 409